

Programación Web

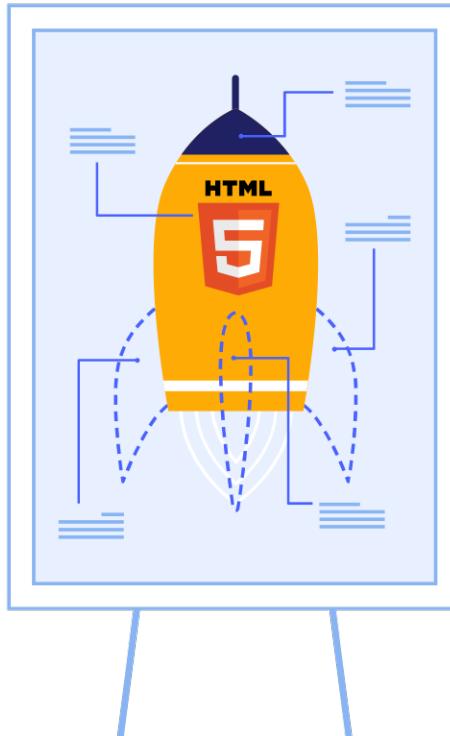

```
}

function getQueue() {
    return queue;
}

function NikeDotcomNavReady(callback) {
    if (dotcomNavInstance) {
        callback(dotcomNavInstance);
    } else if (queue.indexOf(callback) === -1) {
        queue.push(callback);
    }
}
```



<p>



José Luis Quiroz Fabián

Índice

1. JavaScript	4
1.1. ¿Cómo se declara código JavaScript?	4
1.2. Desplegar en consola y mensaje de alerta	4
1.3. Tipos de datos y variables	5
1.4. Funciones	7
1.4.1. Objeto arguments	7
1.4.2. Parámetro por defecto	8
1.4.3. Ejemplo de funciones	8
1.4.4. Recursividad	11
1.4.5. Ejercicio	12
1.5. Estructuras de control	14
1.5.1. Estructuras de condición	14
1.5.2. Estructuras de repetición	17
1.6. Ejercicio	19
1.7. Clases	19
1.7.1. Definición de una clase con class	19
1.7.2. Definición de una clase con function	21
1.7.3. Herencia	23
1.7.4. Sobrecarga	25
1.8. Document Object Model (DOM)	29
1.8.1. Encontrando elemento por id	30
1.8.2. Encontrando elementos por nombre de etiqueta	31
1.8.3. Encontrando elementos por nombre de clase	33
1.8.4. Encontrando elementos selector CSS	34
1.8.5. Encontrando elementos de una colección form	35
1.8.6. Cambiando el contenido de un elemento HTML	36
1.8.7. Cambiando el valor de un atributo	37
1.8.8. Cambiando un estilo	38
1.8.9. Ejercicio: Reloj	41
1.9. Eventos del teclado y del ratón	42
1.10. APIs HTML5	45
1.11. Canvas	46
1.11.1. El área del canvas	46
1.11.2. Una línea en canvas	47
1.11.3. Un círculo en canvas	48
1.11.4. Una animación	49



2. jQuery	49
2.1. ¿Cómo se declara código jQuery?	49
2.2. Sintaxis jQuery	50
2.3. Un tipo de elemento como selector	50
2.4. Un elemento con id como selector	52
2.5. Manejo de eventos	54
2.5.1. Controlando el evento mouseup	54
2.5.2. Entrando y saliendo de un elemento.	56
2.5.3. Múltiples eventos.	59
2.6. Animación.	62
2.7. Agregar nuevos elementos.	65



1 JavaScript

JavaScript es un lenguaje de alto nivel, dinámico, débilmente tipado, orientado a objetos (contiene una biblioteca estándar de objetos como por ejemplo: Array, Date, y Math), multiparadigma e interpretado. Junto con HTML y CSS, JavaScript es una de las tres tecnologías núcleo del contenido World Wide Web. Se utiliza para crear páginas interactivas y proporcionar programas online. Actualmente la mayoría de los sitios lo incluyen, y todos los navegadores modernos lo soportan sin necesidad de incluir un plug-ins lo que significa que mantienen un motor de ejecución JavaScript.

1.1 ¿Cómo se declara código JavaScript?

Puede ser declarando el código directamente:

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <p></p>
5     <script>
6       </script>
7   </body>
8 </html>
```

o mediante referencia a un archivo JavaScript que se declara comúnmente en la etiqueta **head**.

```
1 <script src="myscripts.js"></script>
```

1.2 Desplegar en consola y mensaje de alerta

En JavaScript es muy útil desplegar información en la consola del navegador o mediante una ventana de alerta a fin de depurar nuestro código. Lo anterior se puede realizar de la siguiente forma:

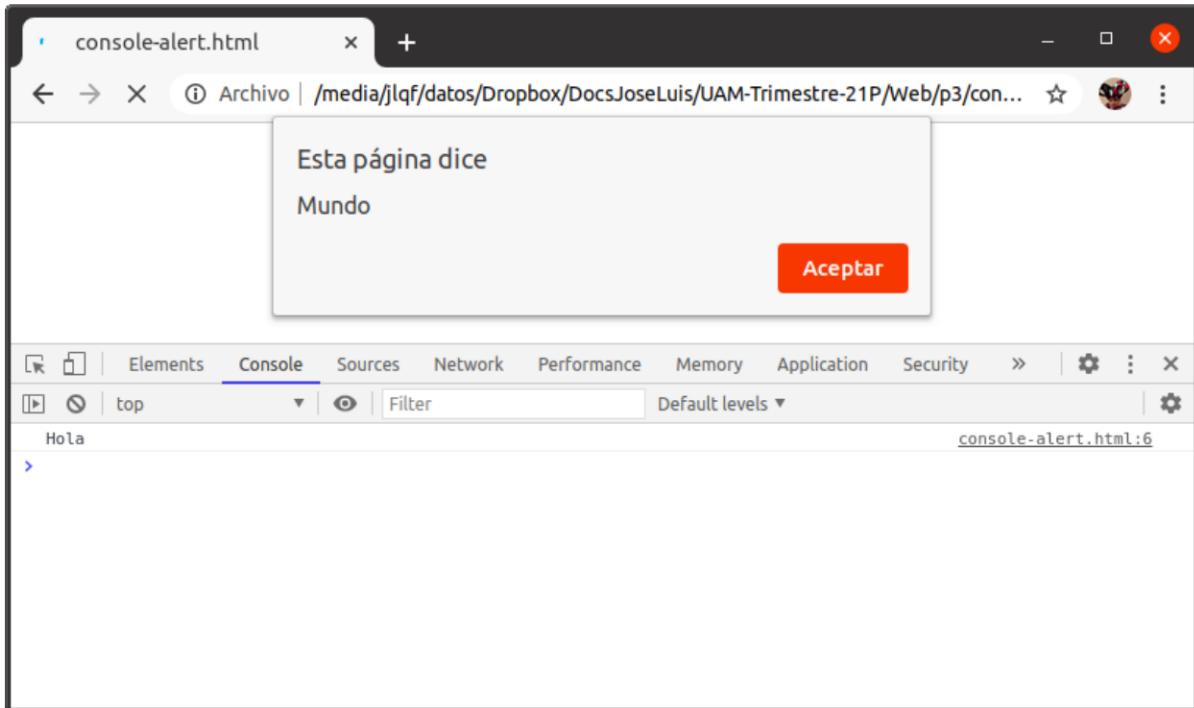
```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <p></p>
5     <script>
6       console.log("Hola");
7       alert("Mundo")
8     </script>
9   </body>
```



10 | </html>

Salida:



1.3 Tipos de datos y variables

JavaScript contempla los siguientes tipos de datos primitivos

1. string.
2. number.
3. boolean.

Los tipos de datos compuestos (de referencia) son:

1. Object
2. Array

Para declarar una variable se utilizan las palabras reservadas:

- **var**: Variables de alcance global.
- **let**: Variables de alcance local.
- **const**: Variable que no puede ser reasignada (constante).



Las variables de JavaScript no tienen ningún tipo predeterminado (como int, double, float, etc), en cambio, el tipo de una variable es el tipo de su valor. Por ejemplo, en el Código 1 la variable *b* tiene originalmente asignado un valor de tipo booleano (línea 8) pero en la línea 13 se le asigna un valor de tipo numérico. La primera mención de la variable la define en la memoria, para que se pueda hacer referencia a ella más adelante en el script.

En JavaScript, se puede realizar operaciones con valores de tipos diferentes sin provocar una excepción. El intérprete de JavaScript convierte implícitamente uno de los tipos de datos en el otro tipo y, después, realiza la operación. Las reglas de conversión de los valores alfanuméricos, numéricos y booleanos son las siguientes:

1. Si se suman un número y una cadena, el número se convierte en una cadena.
2. Si se suman un valor booleano y una cadena, el tipo booleano se convierte en una cadena.
3. Si se suman un número y un valor booleano, el tipo booleano se convierte en un número.

Las variables pueden tomar valores especiales en su inicialización o declaración, los cuales son:

1. null
2. undefined

El primero es para inicializar una variable y el segundo es cuando a una variable no se le asigna un valor inicial.

```

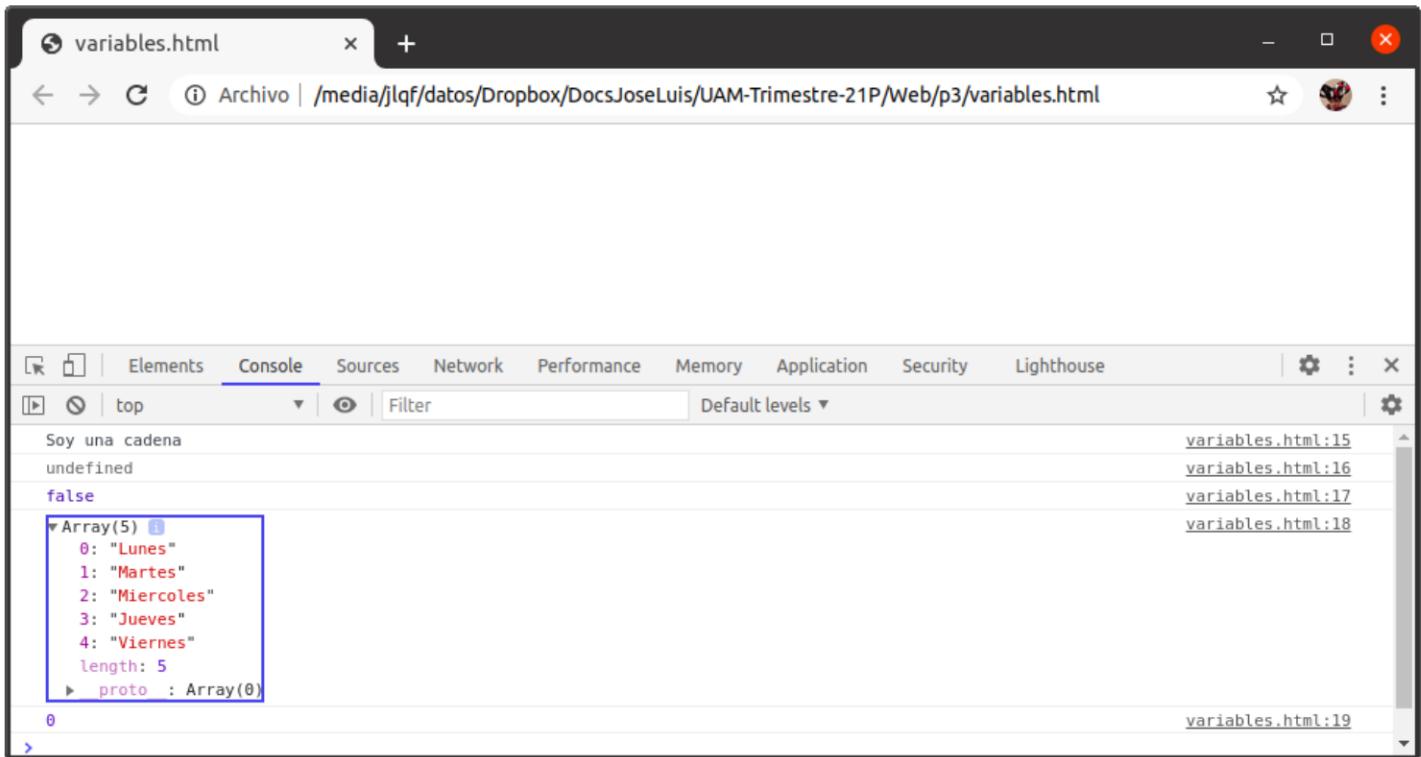
1  <!DOCTYPE html>
2  <html>
3      <body>
4          <p></p>
5          <script>
6              var cad="Soy una cadena";
7              var x;
8              var b=false;
9
10             //Comentario: Arreglo de cadenas
11             dias=["Lunes","Martes","Miercoles","Jueves","Viernes"];
12
13             b=0
14
15             console.log(cad);
16             console.log(x);
17             console.log(b);
18             console.log(dias);
19             console.log(p);
20         </script>
```



```
21   </body>
22 </html>
```

Código 1: Tipos de datos

Salida:



1.4 Funciones

Las funciones en JavaScript se definen mediante la palabra reservada **function**, seguida del nombre de la función. Por ejemplo:

```
1
2 function suma(a,b){
3     c= a+b;
4     return c;
5 }
```

Código 2: Definición de una función

1.4.1 Objeto arguments

Los argumentos de una función son mantenidos en un objeto similar a un array llamado **arguments**. El primer argumento pasado a una función sería **arguments[0]**. El número total de argumentos es mostrado por **arguments.length**.



1.4.2 Parámetro por defecto

En JS los parámetros de una función tienen por defecto un valor de **undefined**, pero se puede establecer un valor por defecto diferente de tal forma que si al invocar a la función no se mandan los parámetros completos estos asumen dicho valor.

1.4.3 Ejemplo de funciones

```

1  <!DOCTYPE html>
2
3  <html>
4      <body>
5          <p></p>
6          <script>
7
8              op1 = prompt("Ingresa un numero");
9              op2 = prompt("Ingresa un numero");
10
11             console.log("Operadores: "+op1+ " , "+op2);
12
13             intOp1=parseInt(op1);
14             intOp2=parseInt(op2);
15
16             r=resta(intOp1,intOp2);
17
18             console.log("Resultado resta:"+r);
19
20             suma(intOp1,intOp2);
21
22             suma("4","6");
23
24             suma(3,4,5);
25
26             multiplicacion();
27
28             multiplicacion(3,4);
29             function resta(a,b){
30                 return a-b;
31             }
32
33             function suma(){
34
35                 console.log("#Argumentos:"+arguments.length)
36                 console.log("Argumento 1: "+arguments[0]);
37                 console.log("Argumento 2: "+arguments[1]);
38                 console.log("Argumento 3: "+arguments[2]);

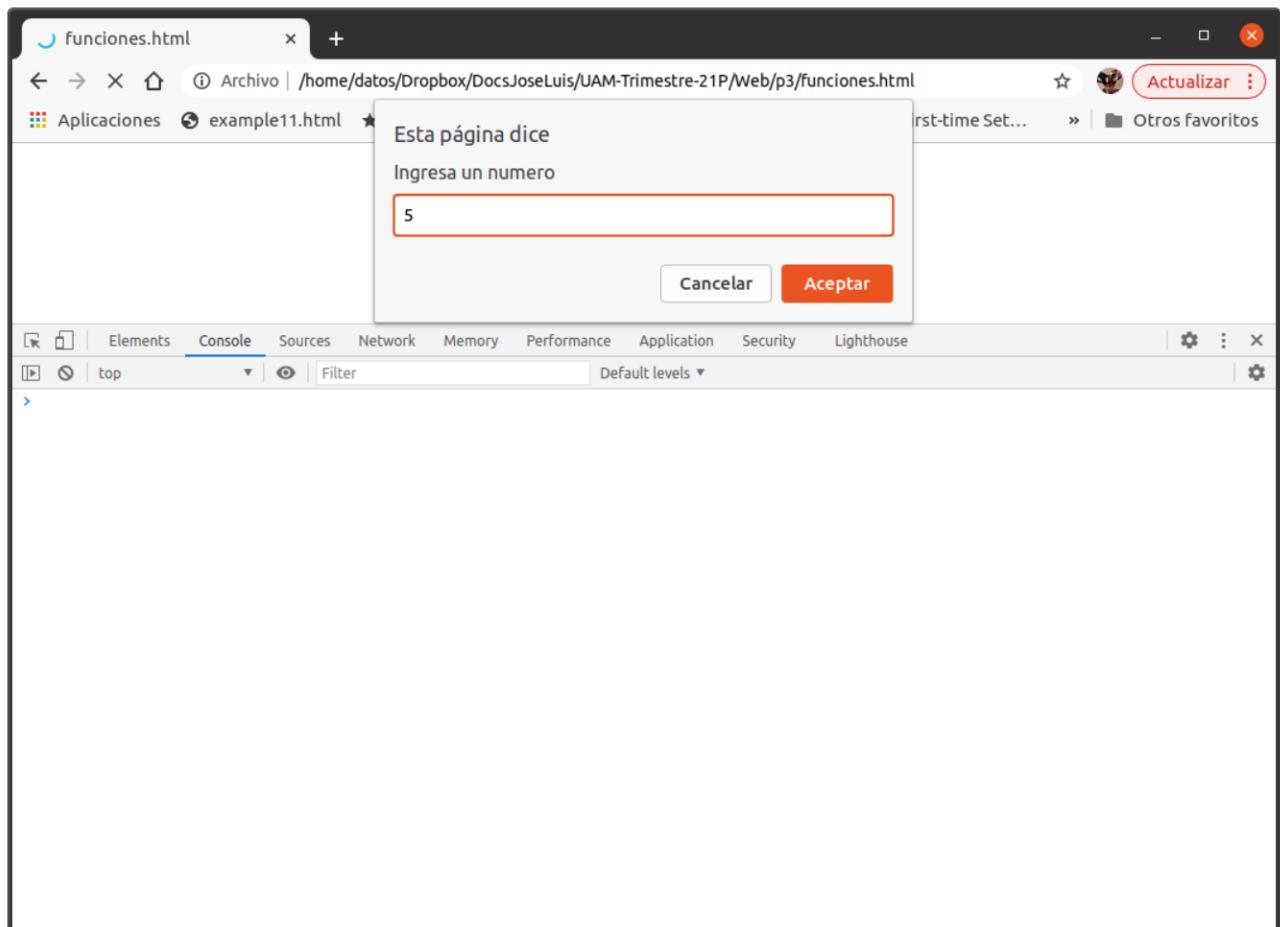
```

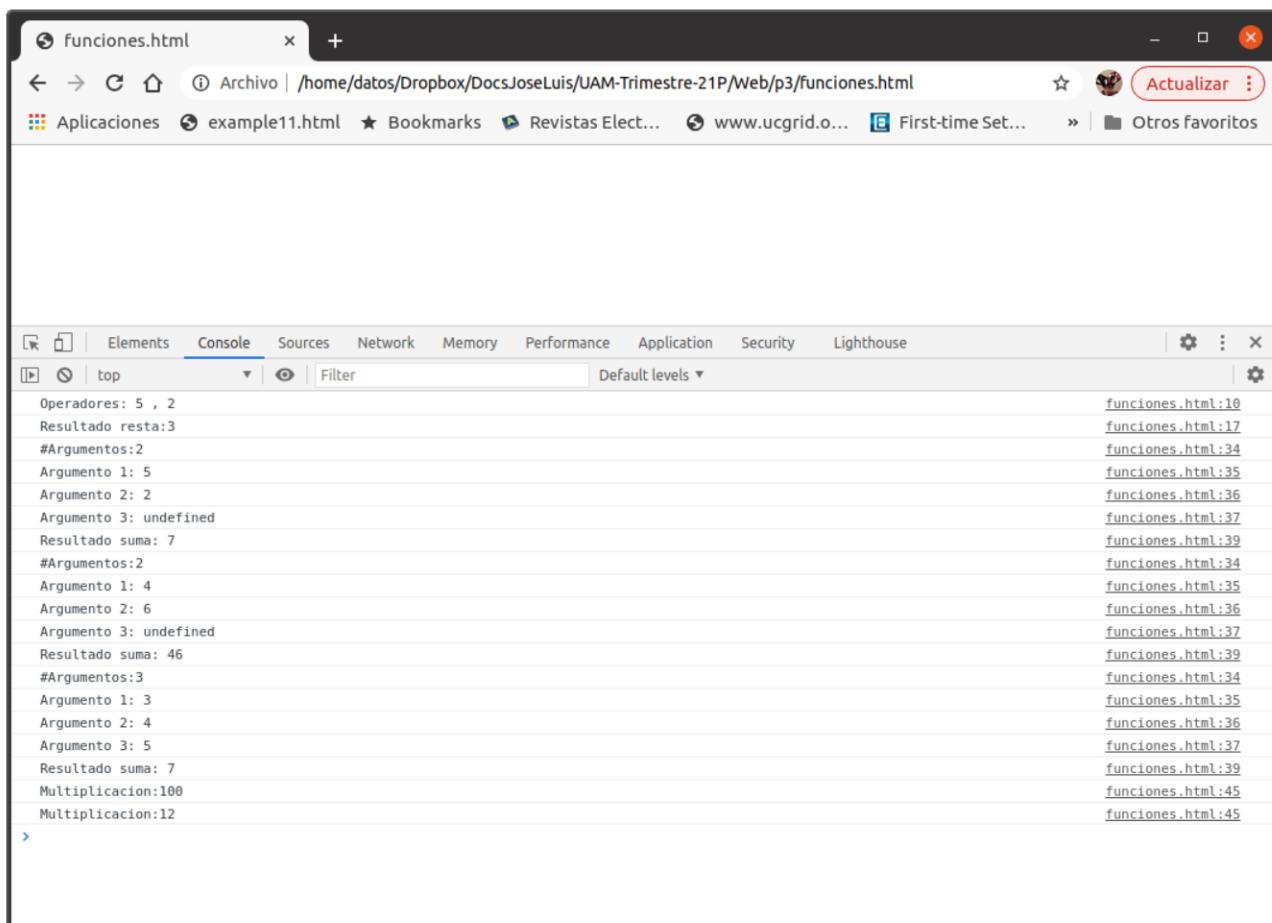
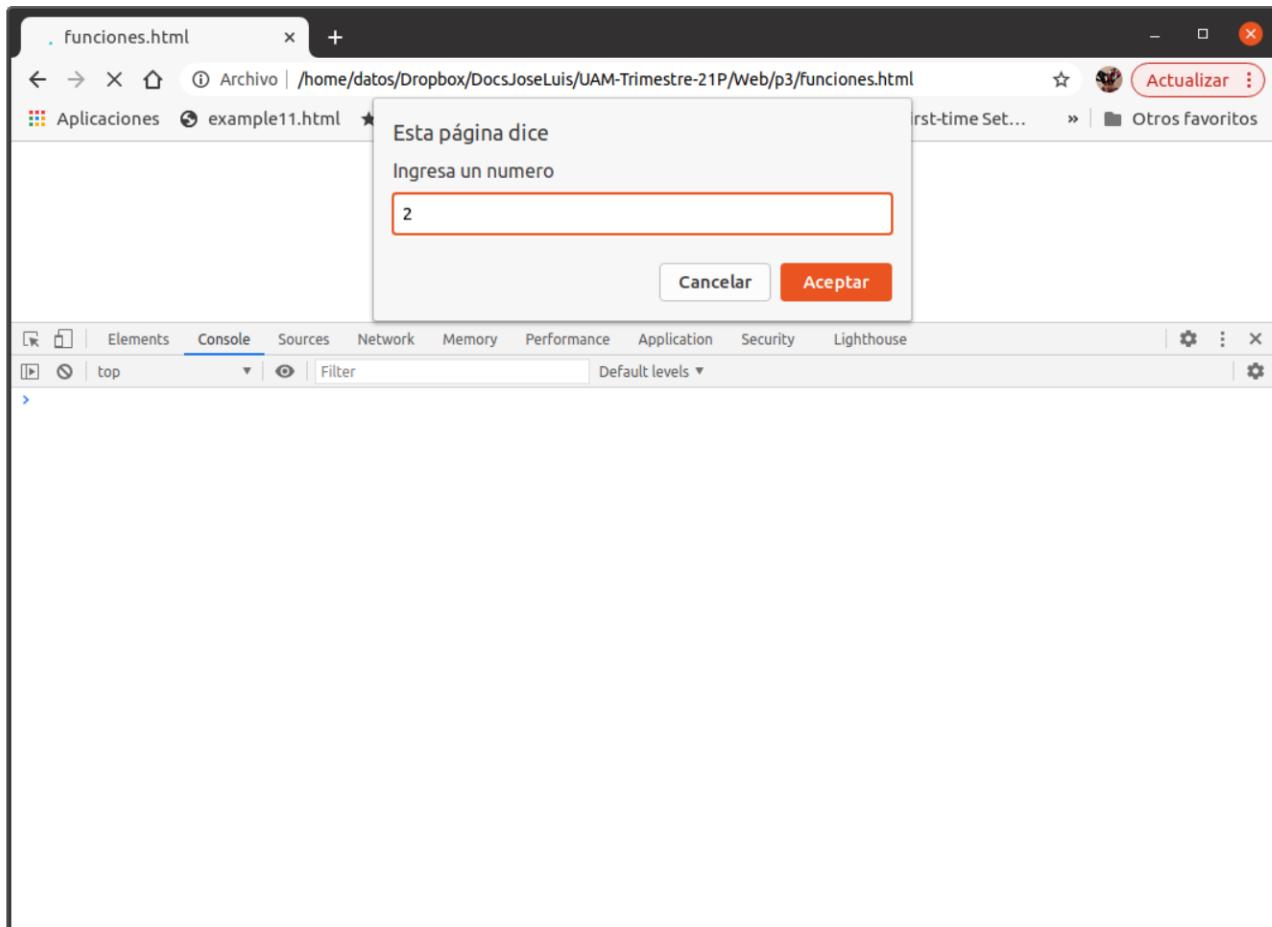


```
39         console.log("Resultado suma: "+(arguments[0]+arguments[1]));
40
41     }
42
43     function multiplicacion(a=5 ,b=20){
44
45         console.log("Multiplicacion: "+a*b);
46
47     }
48
49     </script>
50 </body>
51 </html>
```

Código 3: Ejemplo de funciones

Salida:





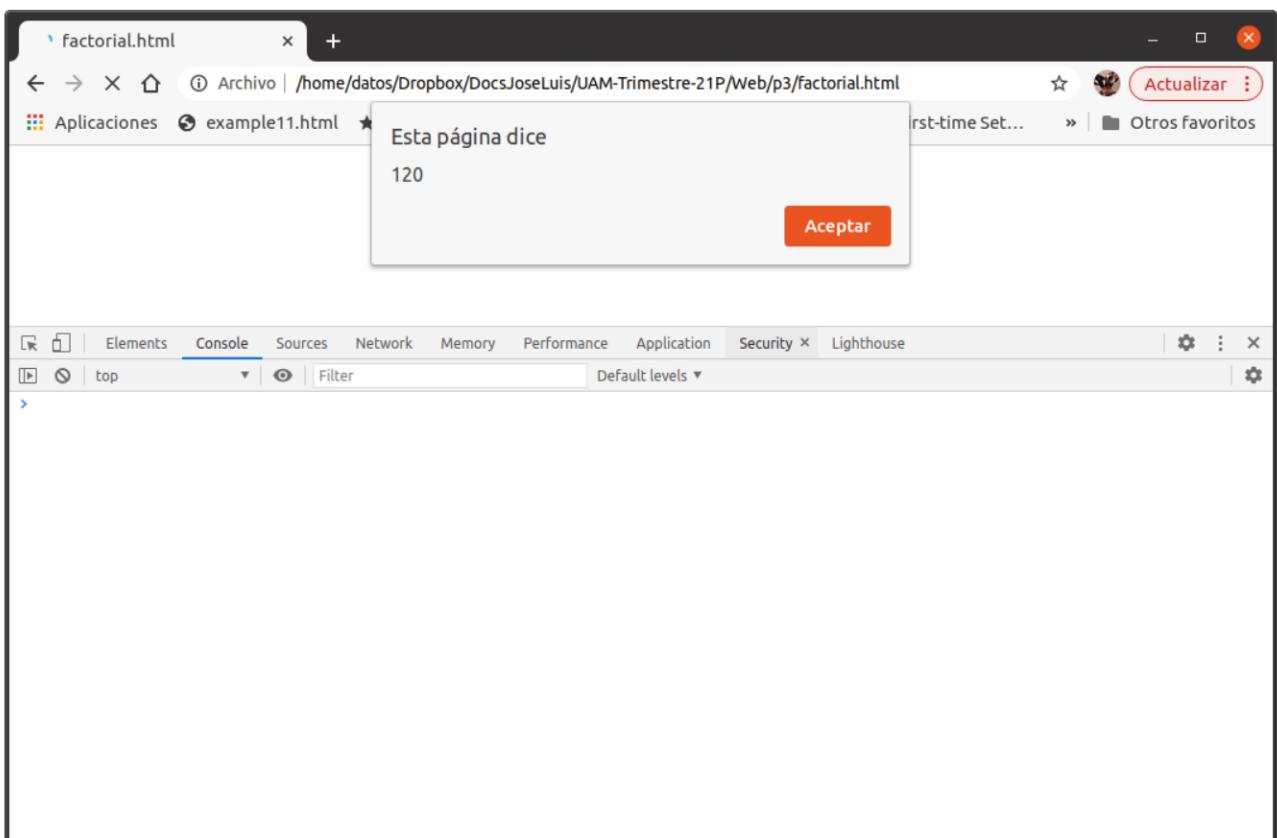
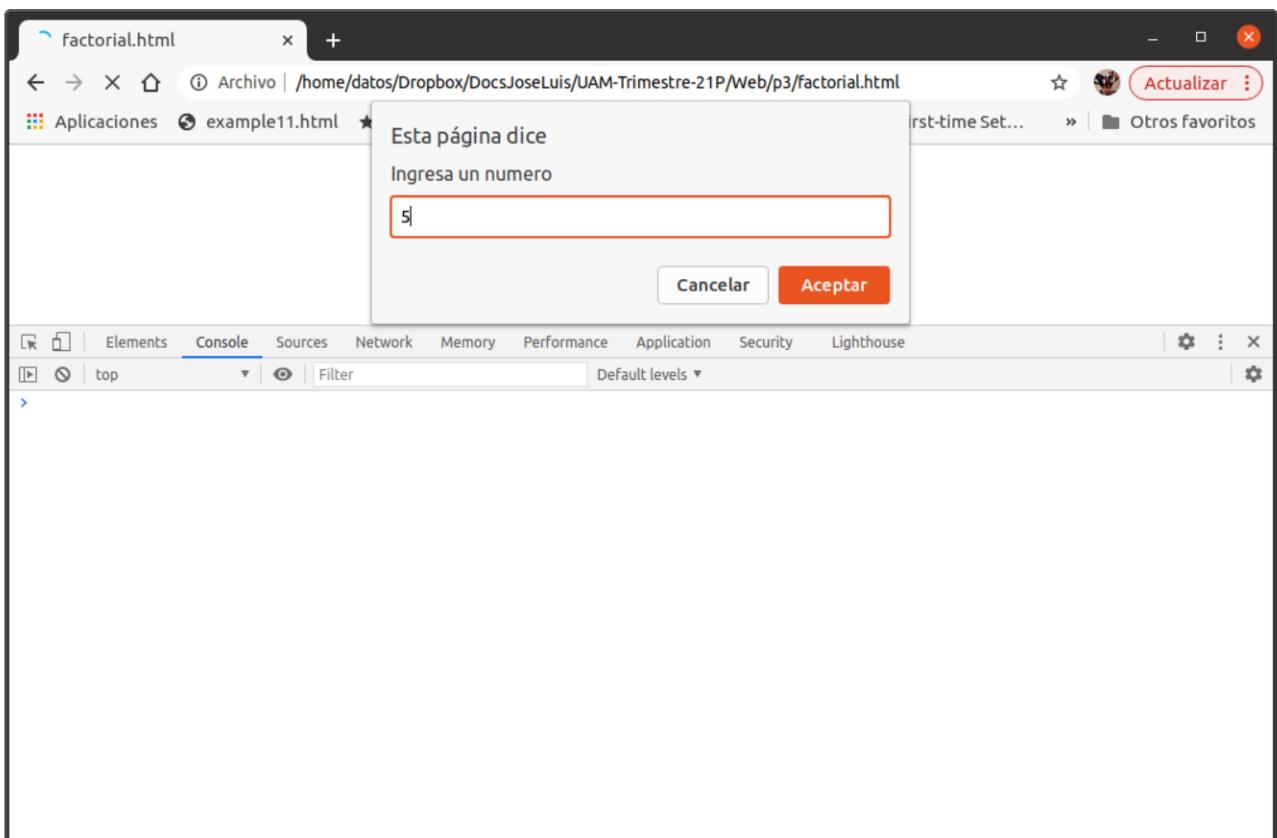
1.4.4 Recursividad

```
1 <!DOCTYPE html>
2 <html>
3   <body>
4     <p></p>
5     <script>
6
7       n = prompt("Ingresa un numero");
8
9       r=factorial(n);
10
11      alert(r);
12
13      function factorial(n){
14        (n==0 || n==1) && (fact=1);
15
16        (n>1) && (fact=n*factorial(n-1));
17
18        return fact;
19
20      }
21
22    </script>
23  </body>
24 </html>
```

Código 4: Ejemplo de funciones

Salida:





1.4.5 Ejercicio

Mediante funciones y sin estructuras condicionales o de repetición calcular la aproximación:



$$e^x \approx \sum_{i=0}^n \frac{(x)^i}{(i!)}$$

Debe ingresar n y x mediante **prompt**.

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <p></p>
5     <script>
6
7       n = prompt("Ingresa n");
8
9       x = prompt("Ingresa x");
10
11      r=aproximacion(parseInt(x),parseInt(n));
12
13      alert(r);
14
15      function factorial(n){
16        (n==0 || n==1) && (fact=1);
17
18        (n>1) && (fact=n*factorial(n-1));
19
20        return fact;
21
22      }
23
24      function potencia(base,exp){
25        (exp==0) && (pot=1);
26
27        (exp==1) && (pot=base);
28
29        (exp>1) && (pot=base*potencia(base,exp-1));
30
31        return pot;
32
33      }
34
35      function aproximacion(x,n){
36        (n==0) && (aprox=1);
37
38        (n>0) && (aprox=potencia(x,n)/factorial(n)+aproximacion(x,n-1));
39
40        return aprox;
41
42      }

```



```

43
44
45     </script>
46 </body>
47 </html>
```

Código 5: Solución del ejercicio

1.5 Estructuras de control

1.5.1 Estructuras de condición

JavaScript soporta las estructuras **if**, **if-else** y **switch** con la sintaxis similar al del lenguaje C y Java.

```

1  <!DOCTYPE html>
2  <html>
3      <body>
4          <p></p>
5          <script>
6
7              function getRomano(){
8                  romano="";
9                  str=document.getElementById("input").value;
10                 valor=parseInt(str);
11
12                 if(valor<0)
13                     alert("El numero debe ser positivo");
14                 else
15                     if(valor>100)
16                         alert("El valor debe estar entre 1 y 100");
17                     else{
18
19                         c=Math.floor(valor/10);
20                         r=valor%10;
21
22                         switch(c){
23                             case 0:
24                                 break;
25                             case 1:
26                                 romano="X";
27                                 break;
28                             case 2:
29                                 romano="XX";
29                                 break;
30                             case 3:
31                                 romano="XXX";
```



```

33                     break;
34         case 4:
35             romano="XL";
36             break;
37         case 5:
38             romano="L";
39             break;
40         case 6:
41             romano="LX";
42             break;
43         case 7:
44             romano="LXX";
45             break;
46         case 8:
47             romano="LXXX";
48             break;
49         case 9:
50             romano="XC";
51     default:
52         romano="C";
53
54     }
55     switch(r){
56         case 0:
57             break;
58         case 1:
59             romano=romano +"I";
60             break;
61         case 2:
62             romano=romano +"II";
63             break;
64         case 3:
65             romano=romano +"III";
66             break;
67         case 4:
68             romano=romano +"IV";
69             break;
70         case 5:
71             romano=romano +"V";
72             break;
73         case 6:
74             romano=romano +"VI";
75             break;
76         case 7:
77             romano=romano +"VII";
78             break;
79         case 8:

```



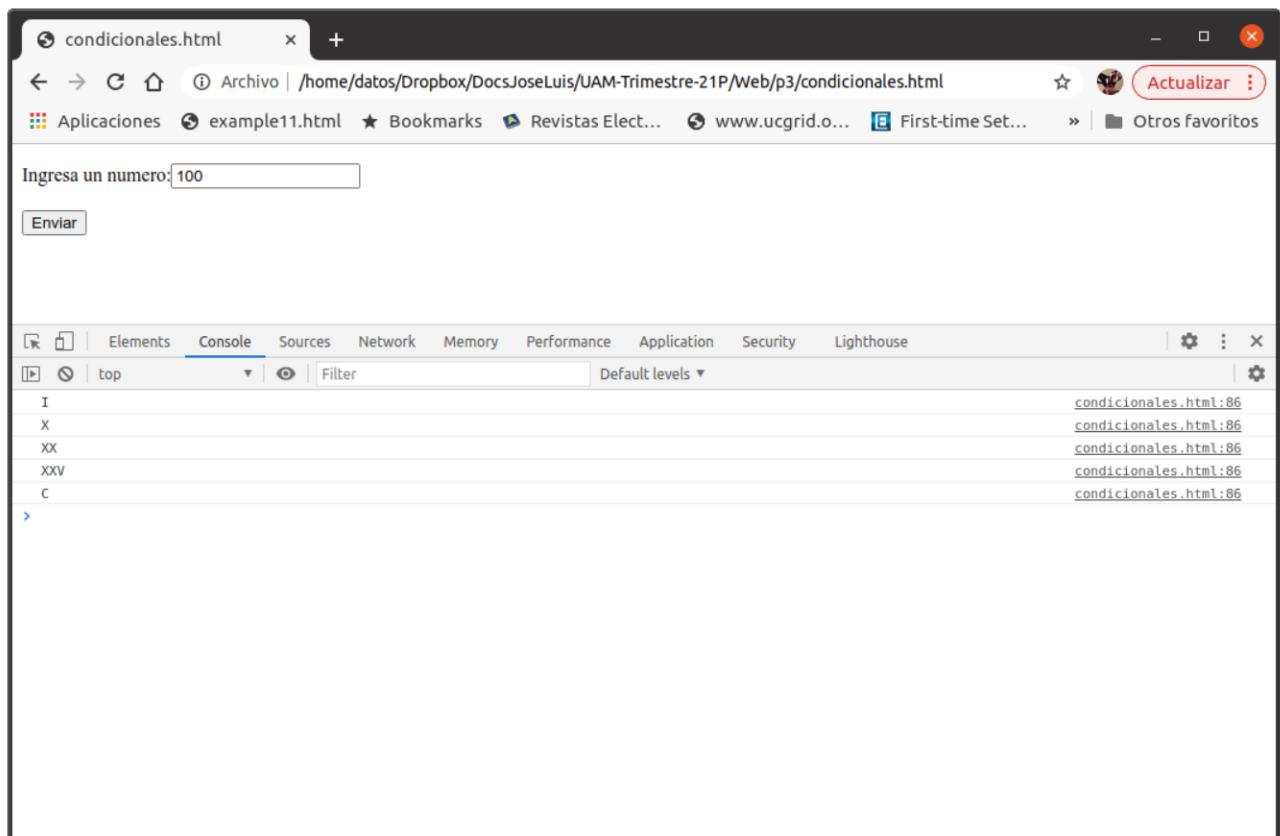
```

80             romano=romano + "VIII";;
81             break;;
82         default:
83             romano=romano + "IX";
84
85     }
86     console.log(romano);
87
88 }
89
90
91 </script>
92
93 <form action="javascript:getRomano()">
94     Ingresa un numero:<input type="number" name="cajas" placeholder="p.e. 45"
95         id="input" required/><br><br>
96     <input type="submit" value="Enviar">
97 </form>
98 </body>
</html>

```

Código 6: Ejemplo de condicionales

Salida:



1.5.2 Estructuras de repetición

JavaScript soporta las estructuras **for**, **while** y **do-while** con la sintaxis similar al del lenguaje C y Java.

```

1  <!DOCTYPE html>
2  <html>
3      <body>
4          <p></p>
5          <script>
6
7
8
9          function getAproximacion(){
10
11              str1=document.getElementById("input1").value;
12              valor1=parseInt(str1);
13              str2=document.getElementById("input2").value;
14              valor2=parseInt(str2);
15
16              r=aproximacion(valor1,valor2);
17
18              document.getElementById("output").value=r;
19
20      }
21
22      function factorial(n){
23          fact=1;
24          for(i=1;i<=n;i++)
25              fact=fact*i;
26          return fact;
27
28      }
29
30      function potencia(base,exp){
31          pot=1;
32          while(exp>0){
33              pot=pot*base;
34              exp--;
35          }
36          return pot;
37
38      }
39
40      function aproximacion(n,x){
41          suma=0;
42          do{

```



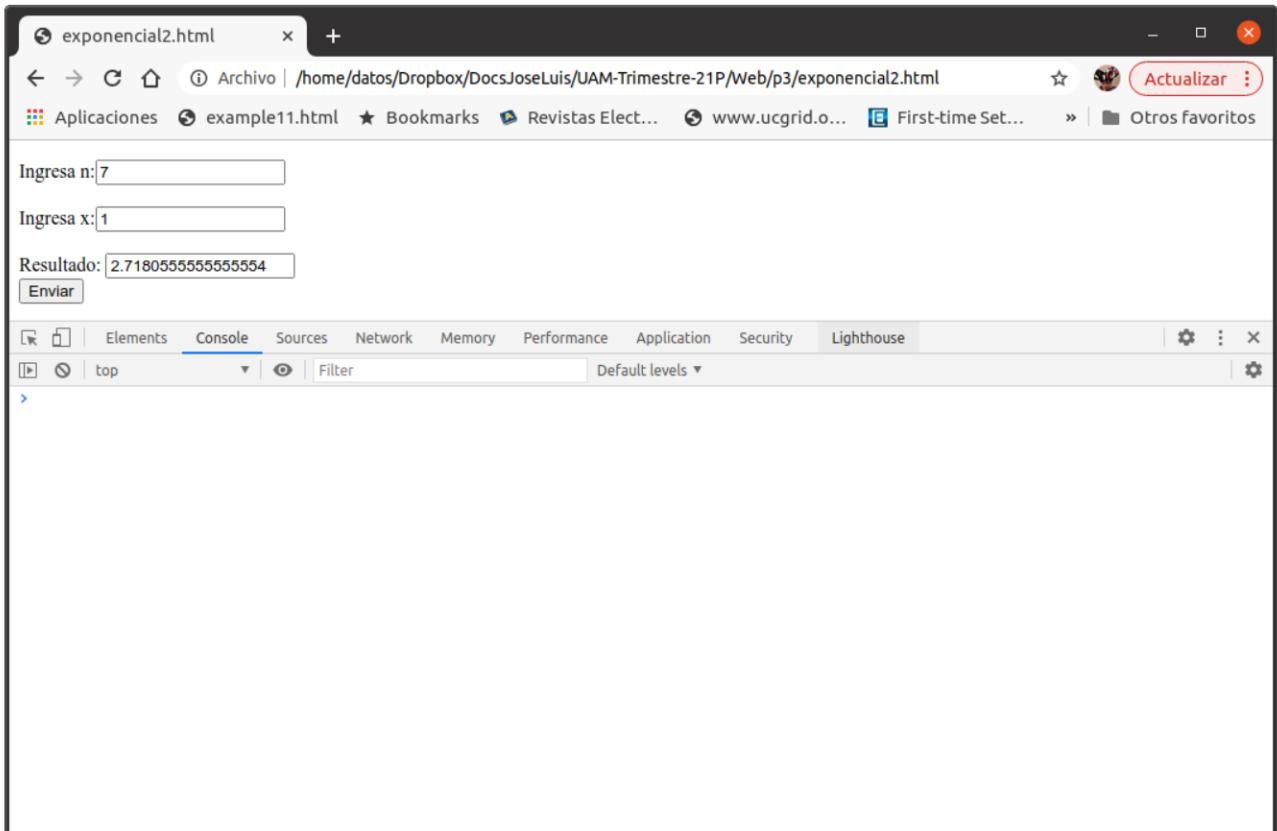
```

43         suma=suma+potencia(n,x)/factorial(n);
44         n--;
45     }
46 }while(n>=0);
47 return suma;
48
49 }
50
51
52 </script>
53 <form action="javascript:getAproximacion()">
54     Ingresa n:<input type="number" name="cajas" placeholder="p.e. 45" id="input1" required/><br><br>
55     Ingresa x:<input type="number" name="cajas" placeholder="p.e. 45" id="input2" required/><br><br>
56     Resultado: <input type="text" name="cajas" value="" id="output" readonly><br>
57     <input type="submit" value="Enviar">
58 </form>
59 </body>
60 </html>

```

Código 7: Ejemplo de estructuras de repetición

Salida:



1.6 Ejercicio

Implementar una función que regrese **true** si todos los símbolos de una cadena se encuentran en otra cadena, **false** en otro caso. Los datos son leídos de cajas de texto y el resultado se despliega en una caja de texto.

1.7 Clases

1.7.1 Definición de una clase con class

Las clases se pueden definir mediante la palabra **class**. La palabra **constructor** define el constructor de la clase, la palabra **this** permite establecer los atributos de la clase. Las palabras **get** y **set** definen atributos *setters* y *getters* para cambiar valores o bien obtener valores de los objetos creados. Al final de la clase se define un método llamado **toString**. Los objetos son creados mediante la palabra **new**.

```

1  <!DOCTYPE html>
2  <html>
3      <body>
4          <p></p>
5          <script>
6
7              var contacto;
8              class Contacto{
9                  constructor(nombre, fecha, correo){
10                      this.nombre=nombre;
11                      this.fecha=fecha;
12                      this.correo=correo;
13                  }
14                  get getNombre(){
15                      return this.nombre;
16                  }
17                  get getFecha(){
18                      return this.fecha;
19                  }
20                  get getCorreo(){
21                      return this.correo;
22                  }
23                  set setNombre(nombre){
24                      this.nombre=nombre;
25                  }
26                  set setFecha(fecha){
27                      this.fecha=fecha;
28                  }
29                  set getCorreo(correo){
30                      this.correo=correo;

```

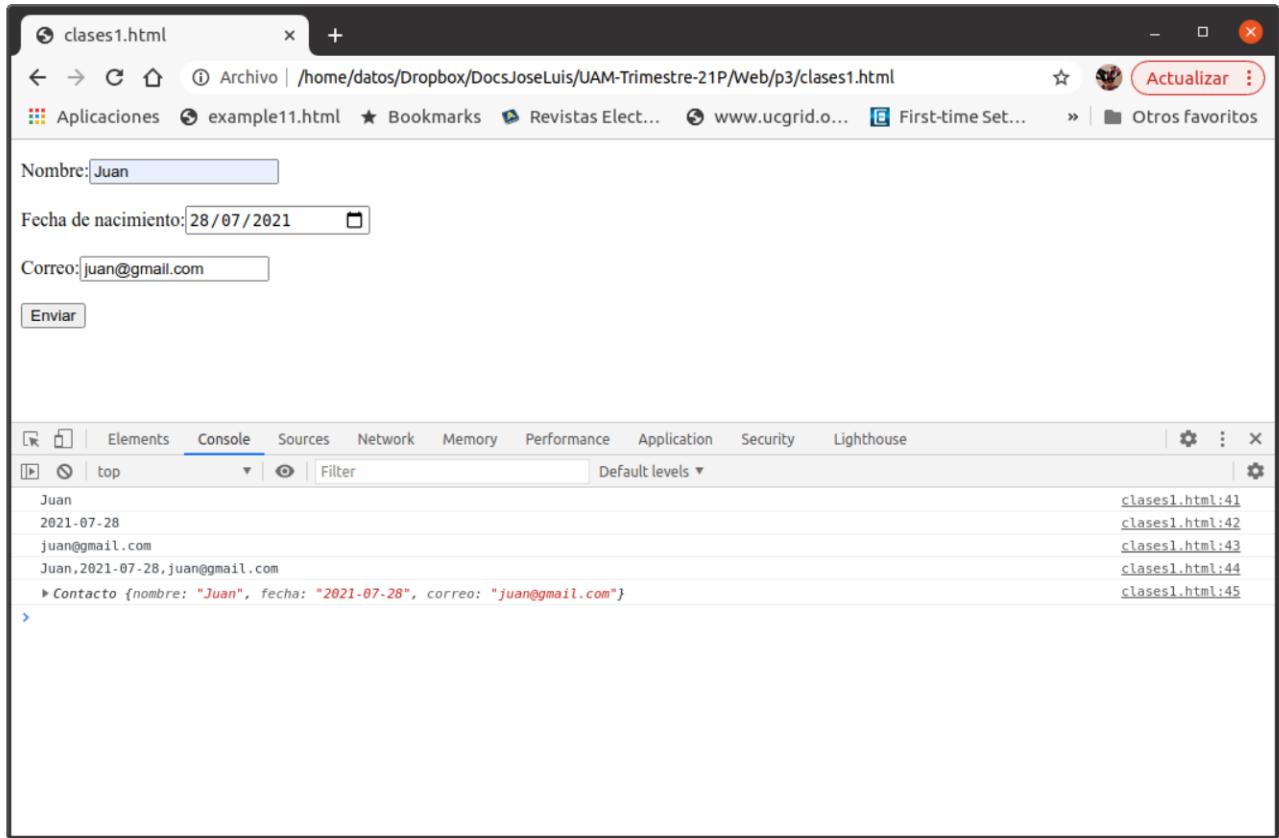


```

31     }
32     toString(){
33         return this.nombre+" "+this.fecha+" "+this.correo;
34     }
35 }
36 function getDatos(){
37     let form = document.getElementById('info');
38     let i;
39     contacto = new Contacto(form[0].value,form[1].value,form[2].value);
40
41     console.log(contacto.getNombre);
42     console.log(contacto.getFecha);
43     console.log(contacto.getCorreo);
44     console.log(contacto.toString());
45     console.log(contacto);
46
47 }
48
49 </script>
50 </body>
51 <form action="javascript:getDatos()" id="info">
52     Nombre:<input type="text" name="cajas" placeholder="p.e. Juan" id="input-nombre" required/><br><br>
53     Fecha de nacimiento:<input type="date" name="cajas" placeholder="p.e. 45" id="input-fecha" required/><br><br>
54     Correo:<input type="email" name="cajas" placeholder="juan@gmail.com" id="input-correo" required/><br><br>
55     <input type="submit" value="Enviar">
56 </form>
57 </html>

```

Código 8: Ejemplo de una clase**Salida:**



1.7.2 Definición de una clase con function

Es posible utilizar **function** para la declaración de una clase. Lo anterior se muestra en el siguiente ejemplo:

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <p></p>
5     <script>
6
7       var contacto;
8       function Contacto(nombre, fecha,correo){
9
10         this.nombre=nombre;
11         this.fecha=fecha;
12         this.correo=correo;
13
14         this.getNombre = function(){
15           return this.nombre;
16         }
17         this.getFecha = function(){
18           return this.fecha;
19         }
20         this.getCorreo = function(){
21           return this.correo;

```



```

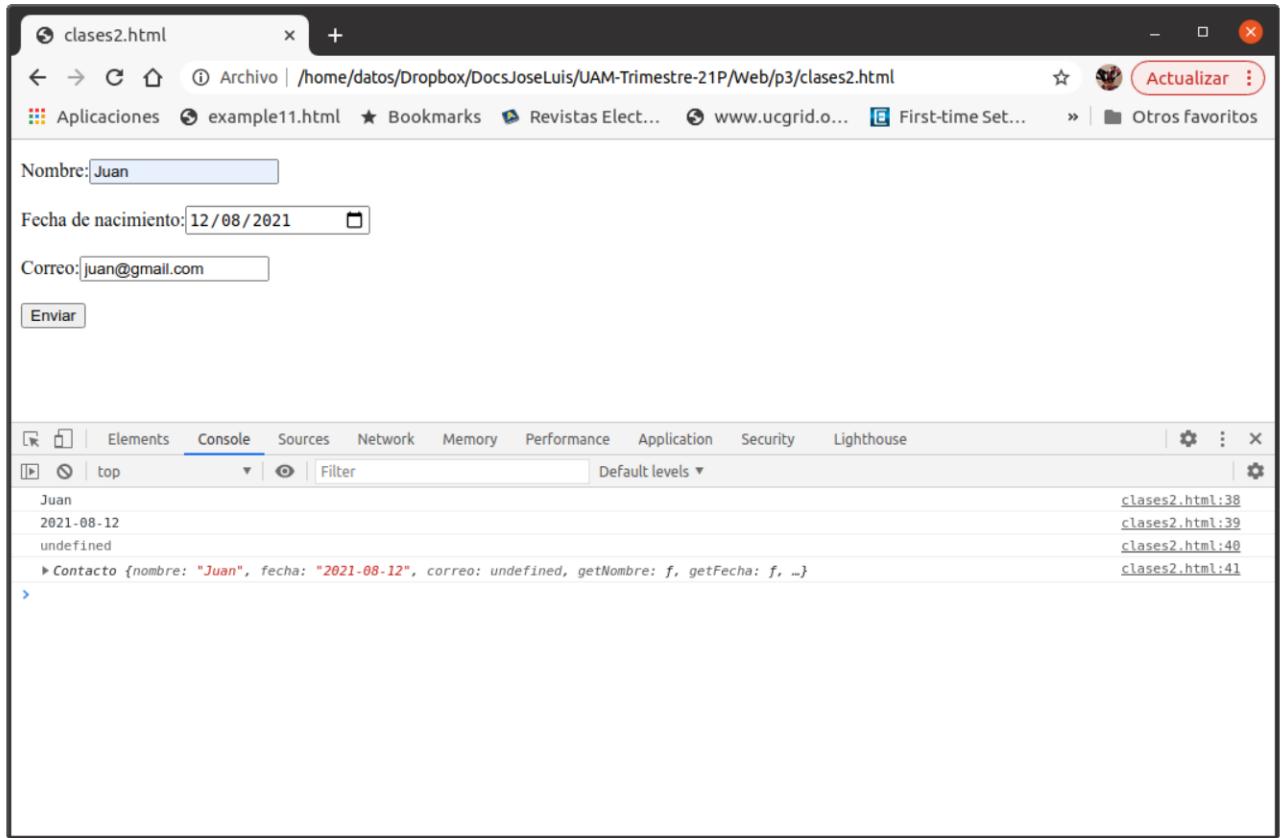
22         }
23     this.setNombre = function(nombre){
24         this.nombre=nombre;
25     }
26     this.setFecha = function(fecha){
27         this.fecha=fecha;
28     }
29     this.getCorreo = function(correo){
30         this.correo=correo;
31     }
32 }
33 function getDatos(){
34     let form = document.getElementById('info');
35     let i;
36     contacto = new Contacto(form[0].value,form[1].value,form[2].value);
37
38     console.log(contacto.getNombre());
39     console.log(contacto.getFecha());
40     console.log(contacto.getCorreo());
41     console.log(contacto);
42
43 }
44
45 </script>
46 </body>
47 <form action="javascript:getDatos()" id="info">
48     Nombre:<input type="text" name="cajas" placeholder="p.e. Juan" id="input-nombre" required/><br><br>
49     Fecha de nacimiento:<input type="date" name="cajas" placeholder="p.e. 45" id="input-fecha" required/><br><br>
50     Correo:<input type="email" name="cajas" placeholder="juan@gmail.com" id="input-correo" required/><br><br>
51     <input type="submit" value="Enviar">
52 </form>
53 </html>

```

Código 9: Ejemplo de una clase con function

Salida:





1.7.3 Herencia

La herencia es posible mediante **extends**. La palabra **super** permite hacer referencia a la clase padre.

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <p></p>
5     <script>
6
7       var contacto;
8       class Persona{
9         constructor(nombre, fecha){
10           this.nombre=nombre;
11           this.fecha=fecha;
12         }
13         get getNombre(){
14           return this.nombre;
15         }
16         get getFecha(){
17           return this.fecha;
18         }
19         set setNombre(nombre){
20           this.nombre=nombre;
21         }

```



```

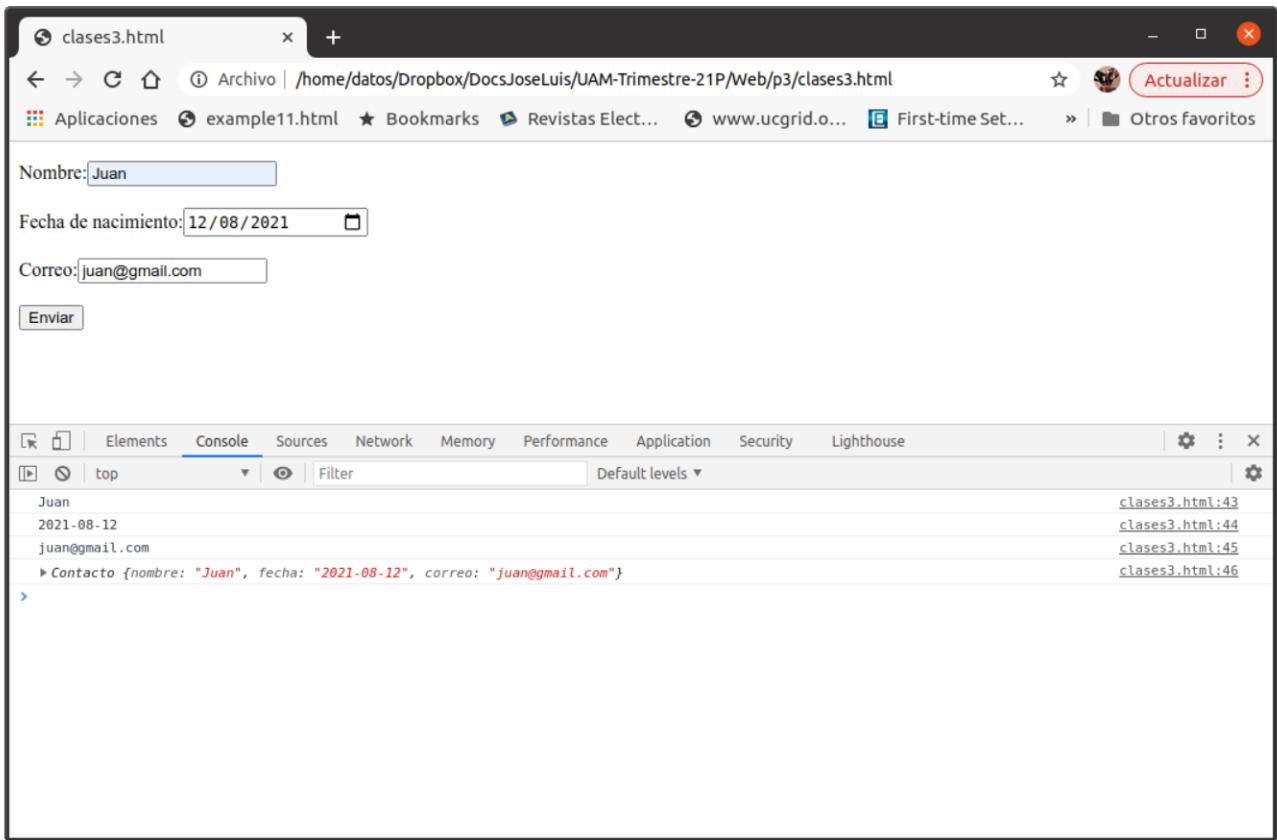
22         set setFecha(fecha){
23             this.fecha=fecha;
24         }
25     }
26     class Contacto extends Persona{
27         constructor(nombre, fecha, correo){
28             super(nombre,fecha);
29             this.correo=correo;
30         }
31         get getCorreo(){
32             return this.correo;
33         }
34         set setCorreo(correo){
35             this.correo=correo;
36         }
37     }
38     function getDatos(){
39         let form = document.getElementById('info');
40         let i;
41         contacto = new Contacto(form[0].value,form[1].value,form[2].value);
42
43         console.log(contacto.getNombre());
44         console.log(contacto.getFecha());
45         console.log(contacto.getCorreo());
46         console.log(contacto);
47
48     }
49
50     </script>
51 </body>
52 <form action="javascript:getDatos()" id="info">
53     Nombre:<input type="text" name="cajas" placeholder="p.e. Juan" id="input-nombre" required/><br><br>
54     Fecha de nacimiento:<input type="date" name="cajas" placeholder="p.e. 45" id="input-fecha" required/><br><br>
55     Correo:<input type="email" name="cajas" placeholder="juan@gmail.com" id="input-correo" required/><br><br>
56     <input type="submit" value="Enviar">
57 </form>
58 </html>

```

Código 10: Herencia

Salida:





1.7.4 Sobrecarga

La sobrecarga es posible aprovechando la palabra **undefined**. Se tiene que verificar que parámetros tienen valor **undefined** para la toma de decisiones del valor de los atributos.

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <p></p>
5     <script>
6
7       var contacto;
8       class Persona{
9         constructor(nombre, fecha){
10           this.nombre=nombre;
11           this.fecha=fecha;
12         }
13         get getNombre(){
14           return this.nombre;
15         }
16         get getFecha(){
17           return this.fecha;
18         }
19         set setNombre(nombre){
20           this.nombre=nombre;
21         }

```



```

22     set setFecha(fecha){
23         this.fecha=fecha;
24     }
25 }
26 class Contacto extends Persona{
27     constructor(nombre, fecha, correo){
28         if(correo==undefined){
29             correo="Sin correo";
30         }
31         if(fecha==undefined){
32             fecha="Sin fecha";
33         }
34         if(nombre==undefined){
35             nombre="Sin nombre";
36         }
37         super(nombre,fecha);
38         this.correo=correo;
39     }
40     get getCorreo(){
41         return this.correo;
42     }
43     set getCorreo(correo){
44         this.correo=correo;
45     }
46 }
47 function getDatos(){
48     let form = document.getElementById('info');
49     let i;
50     contacto = new Contacto();
51
52     console.log(contacto.getNombre);
53     console.log(contacto.getFecha);
54     console.log(contacto.getCorreo);
55
56     contacto = new Contacto(form[0].value,form[1].value,form[2].value);
57
58     console.log(contacto.getNombre);
59     console.log(contacto.getFecha);
60     console.log(contacto.getCorreo);
61
62     contacto = new Contacto(form[0].value,form[1].value);
63
64     console.log(contacto.getNombre);
65     console.log(contacto.getFecha);
66     console.log(contacto.getCorreo);
67 }
68

```



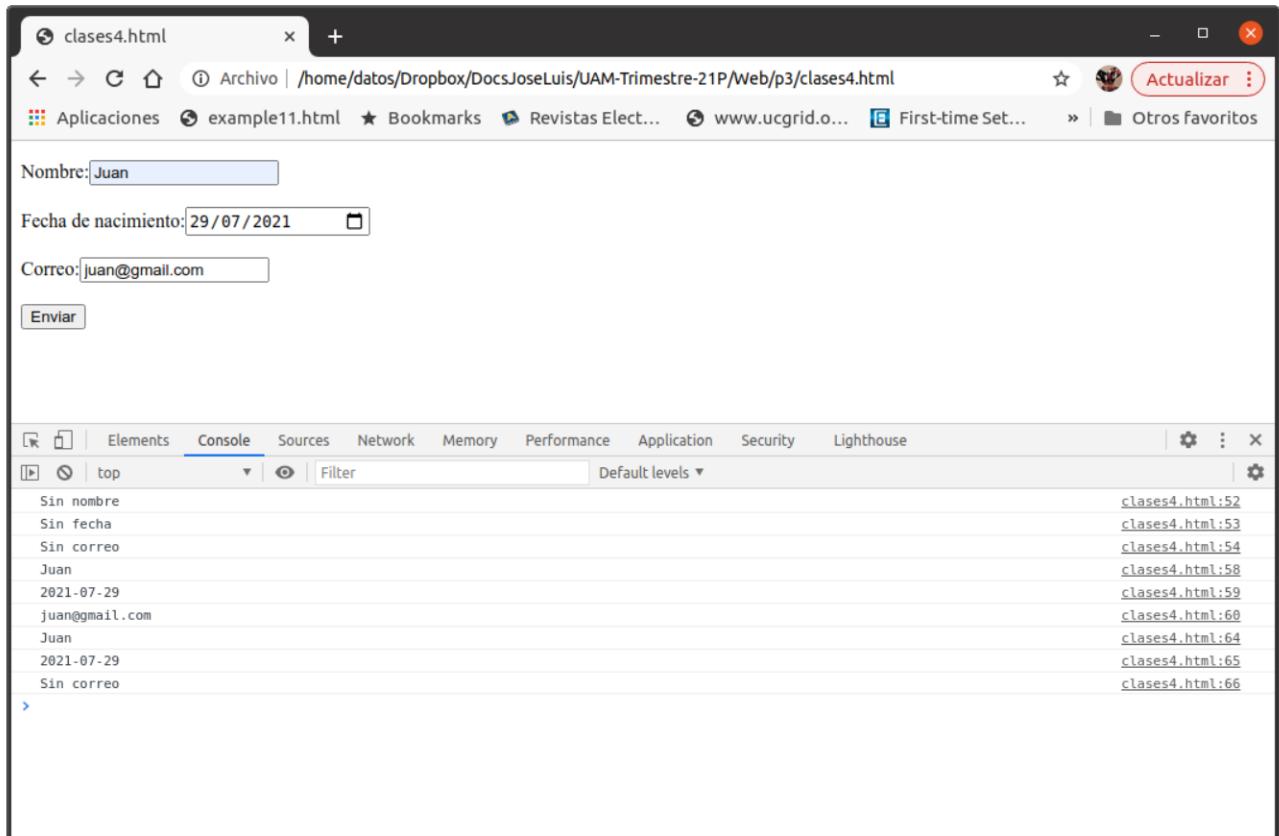
```

69      </script>
70  </body>
71  <form action="javascript:getDatos();" id="info">
72      Nombre:<input type="text" name="cajas" placeholder="p.e. Juan" id="input-nombre"/><br><br>
73      Fecha de nacimiento:<input type="date" name="cajas" placeholder="p.e. 45" id="input-fecha"/><br><br>
74      Correo:<input type="email" name="cajas" placeholder="juan@gmail.com" id="input-correo"/><br><br>
75      <input type="submit" value="Enviar">
76  </form>
77 </html>

```

Código 11: Sobrecarga

Salida:



```

1  <!DOCTYPE html>
2  <html>
3      <body>
4          <p></p>
5          <script>
6
7              var contacto;
8              class Persona{
9                  constructor(nombre, fecha){

```



```

10         this.nombre=nombre;
11         this.fecha=fecha;
12     }
13     get getNombre(){
14         return this.nombre;
15     }
16     get getFecha(){
17         return this.fecha;
18     }
19     set setNombre(nombre){
20         this.nombre=nombre;
21     }
22     set setFecha(fecha){
23         this.fecha=fecha;
24     }
25 }
26 class Contacto extends Persona{
27     constructor(nombre="Sin nombre", fecha="Sin fecha",correo="Sin
28         correo"){
29
30         super(nombre,fecha);
31         this.correo=correo;
32     }
33     get getCorreo(){
34         return this.correo;
35     }
36     set getCorreo(correo){
37         this.correo=correo;
38     }
39     function getDatos(){
40         let form = document.getElementById('info');
41         let i;
42
43         contacto = new Contacto(form[0].value,form[1].value);
44
45         console.log(contacto.getNombre());
46         console.log(contacto.getFecha());
47         console.log(contacto.getCorreo());
48     }
49
50     </script>
51 </body>
52 <form action="javascript:getDatos()" id="info">
53     Nombre:<input type="text" name="cajas" placeholder="p.e. Juan" id="
54         input-nombre"/><br><br>
      Fecha de nacimiento:<input type="date" name="cajas" placeholder="p.e. 45"

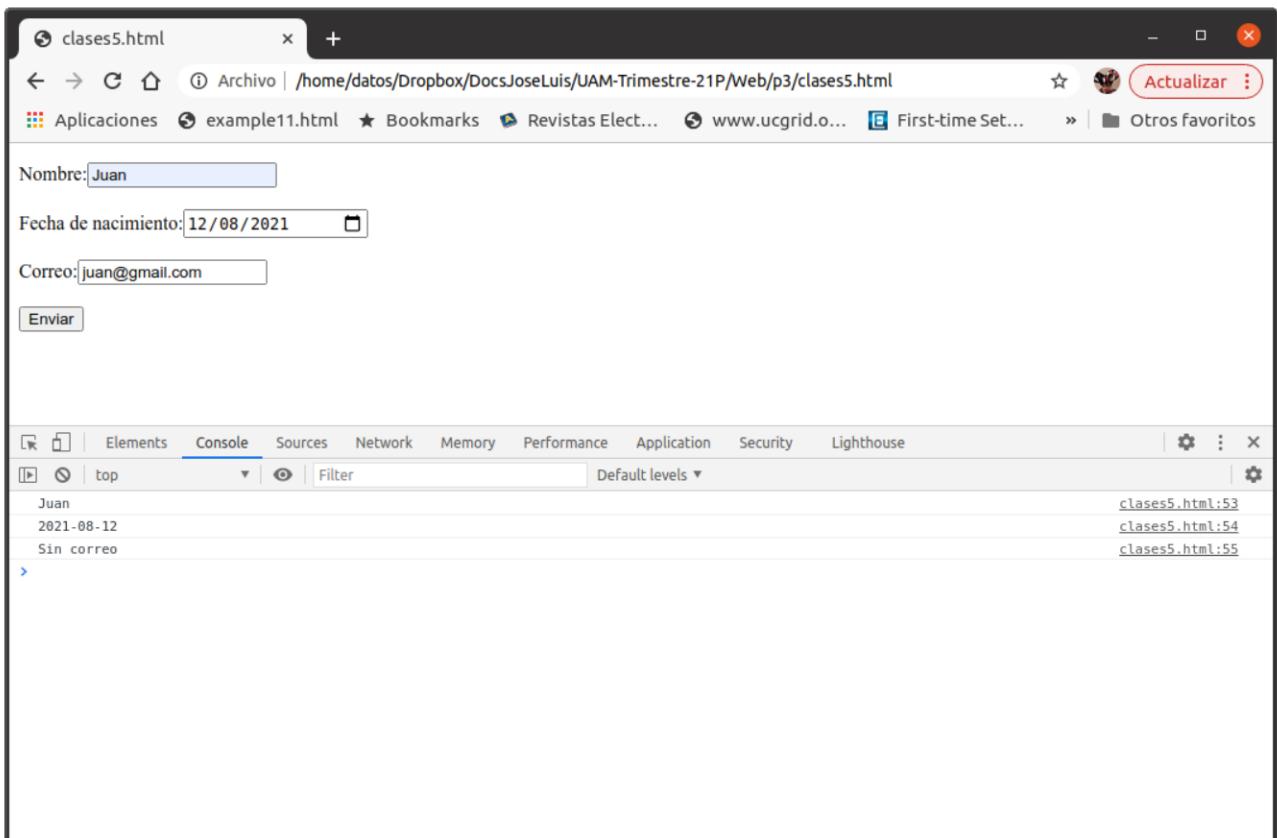
```



```

55     id="input-fecha"/><br><br>
56     Correo:<input type="email" name="cajas" placeholder="juan@gmail.com" id="input-correo"/><br><br>
57     <input type="submit" value="Enviar">
58 </form>
</html>

```

Código 12: Sobrecarga inicializando los parámetros**Salida:**

1.8 Document Object Model (DOM)

El DOM es un estándar W3C (World Wide Web Consortium) para acceder a documentos:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

El estándar W3C DOM es separado en tres partes diferentes:

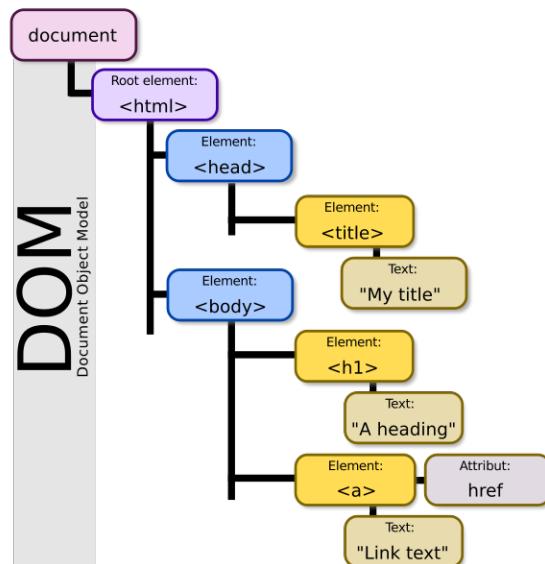
1. Core DOM - modelo estándar para todo tipo de documentos.
2. XML DOM - modelo estándar para documentos XML.
3. HTML DOM - modelo estándar para documentos HTML.



Cuando una página se carga, el navegador crea un documento de modelo de objetos. El modelo DOM se construye como un árbol de objetos. Con el modelo de objetos, JavaScript puede crear páginas HTML dinámicas.

¿Qué operaciones puede realizar JS con los elementos del DOM?

1. JavaScript puede cambiar todos los elementos HTML en la página
2. JavaScript puede cambiar todos los atributos HTML en la página
3. JavaScript puede cambiar todos los estilos CSS en la página
4. JavaScript puede eliminar elementos y atributos HTML
5. JavaScript puede agregar nuevos elementos HTML y atributos
6. JavaScript puede escuchar de todos los eventos en la página
7. JavaScript puede crear nuevos eventos HTML en la página.



1.8.1 Encontrando elemento por id

```

1  <!DOCTYPE html>
2  <html>
3      <body>
4
5          <p id="intro">Hola Mundo!</p>
6          <p>Obtiene un elemento por el metodo <b>getElementById</b></p>
7          <p id="demo"></p>
8          <script>
9              var elemento = document.getElementById("intro");
  
```



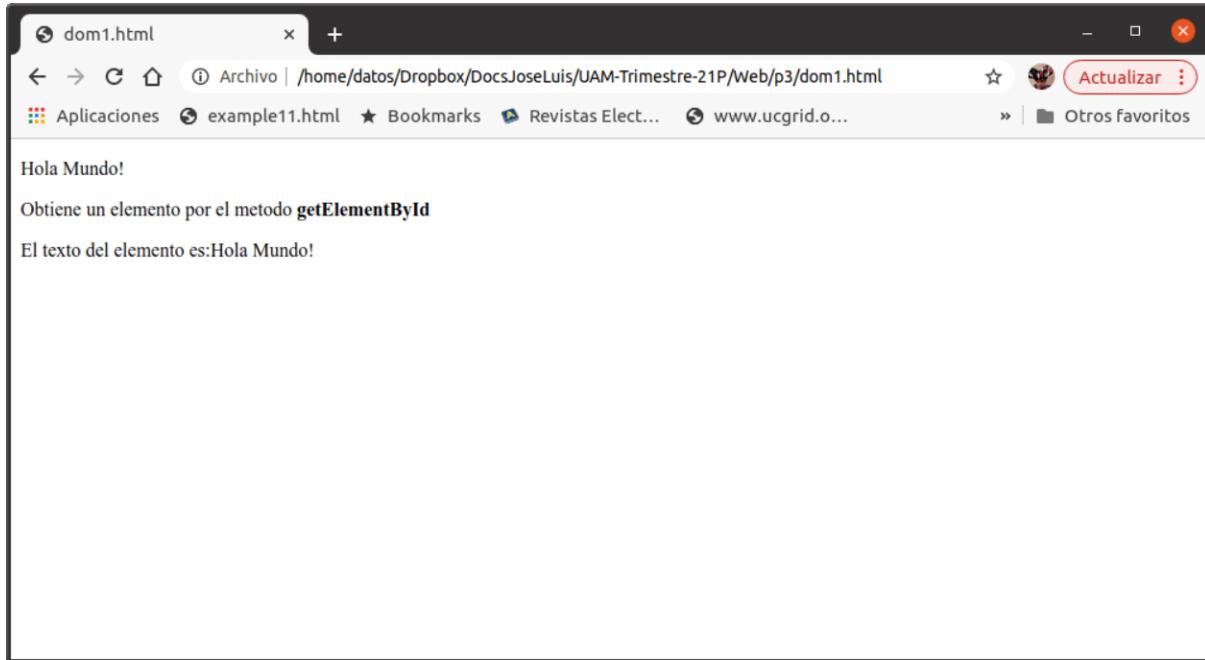
```

10     document.getElementById("demo").innerHTML = "El texto del elemento es:"
11         + elemento.innerHTML;
12     </script>
13 
14 </body>
</html>

```

Código 13: Encontrar elemento por id

Salida:



1.8.2 Encontrando elementos por nombre de etiqueta

```

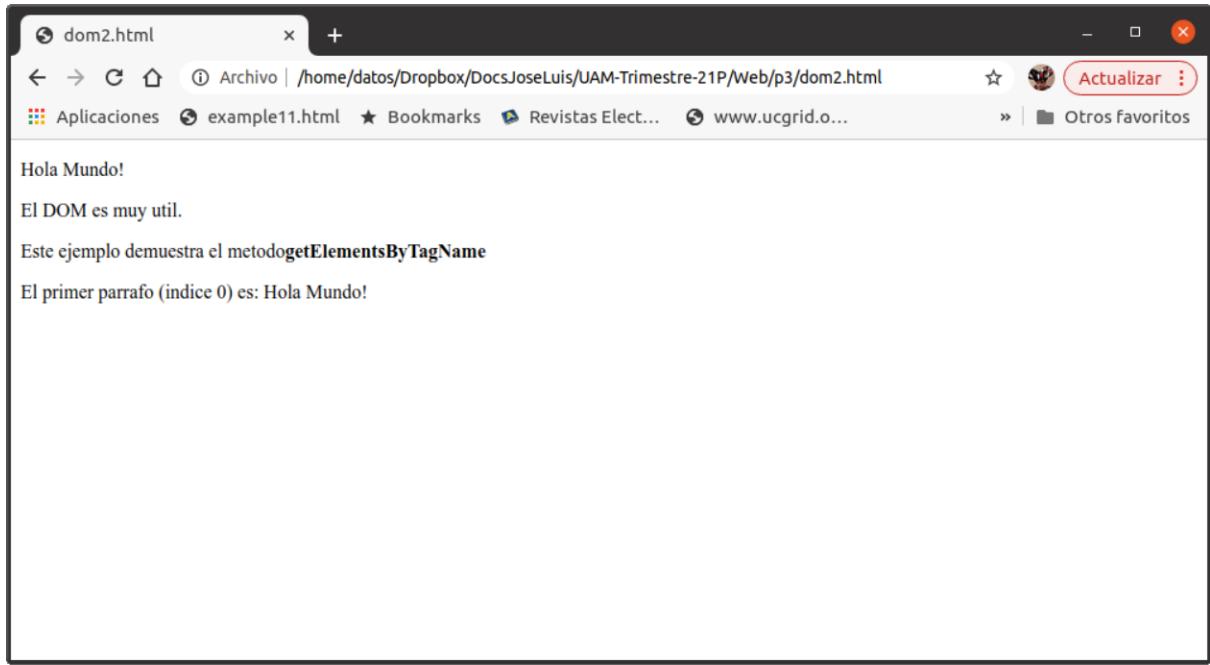
1 <!DOCTYPE html>
2 <html>
3   <body>
4     <p>Hola Mundo!</p>
5     <p>El DOM es muy util.</p>
6     <p>Este ejemplo demuestra el metodo<b>getElementsByTagName</b> </p>
7
8     <p id="demo"></p>
9
10    <script>
11      var x = document.getElementsByTagName("p");
12      document.getElementById("demo").innerHTML = 'El primer parrafo (indice
13          0) es: ' + x[0].innerHTML;
14    </script>
15  </body>
</html>

```

Código 14: Encontrar elemento por etiqueta



Salida:



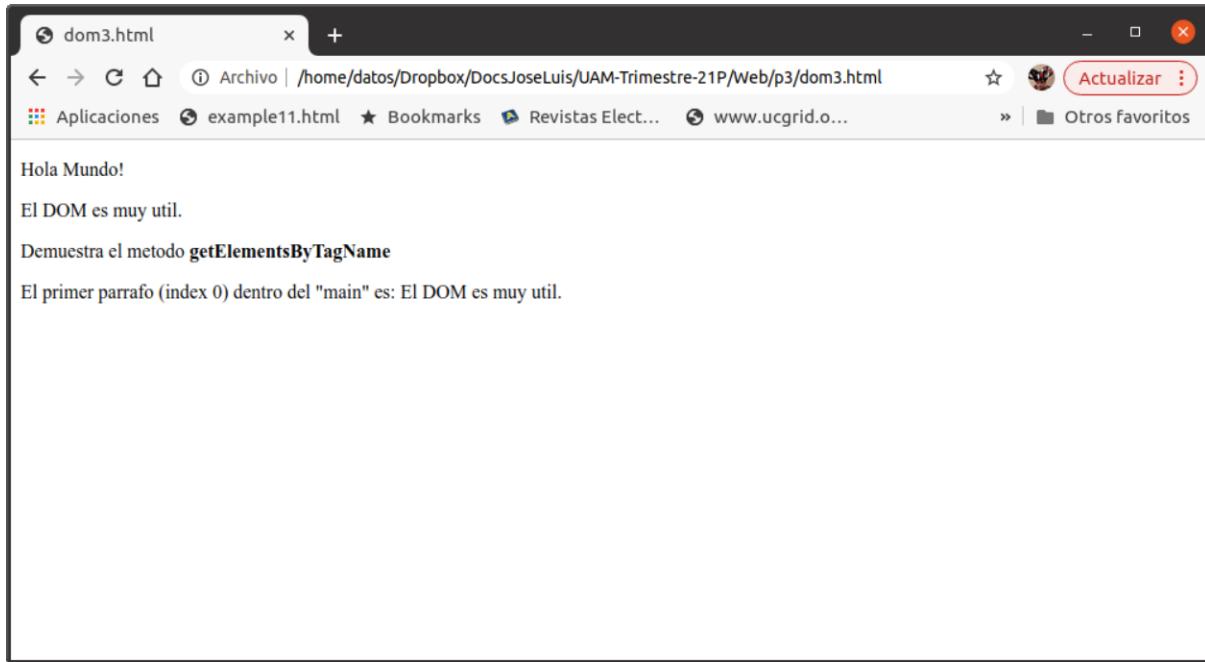
```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <p>Hola Mundo!</p>
5     <div id="main">
6       <p>El DOM es muy util.</p>
7       <p>Demuestra el metodo <b>getElementsByTagName</b></p>
8     </div>
9     <p id="demo"></p>
10    <script>
11      var x = document.getElementById("main");
12      var y = x.getElementsByTagName("p");
13      document.getElementById("demo").innerHTML = 'El primer parrafo (index
14        0) dentro del "main" es: ' + y[0].innerHTML;
15    </script>
16  </body>
</html>
```

Código 15: Encontrar elemento por etiqueta

Salida:





1.8.3 Encontrando elementos por nombre de clase

```

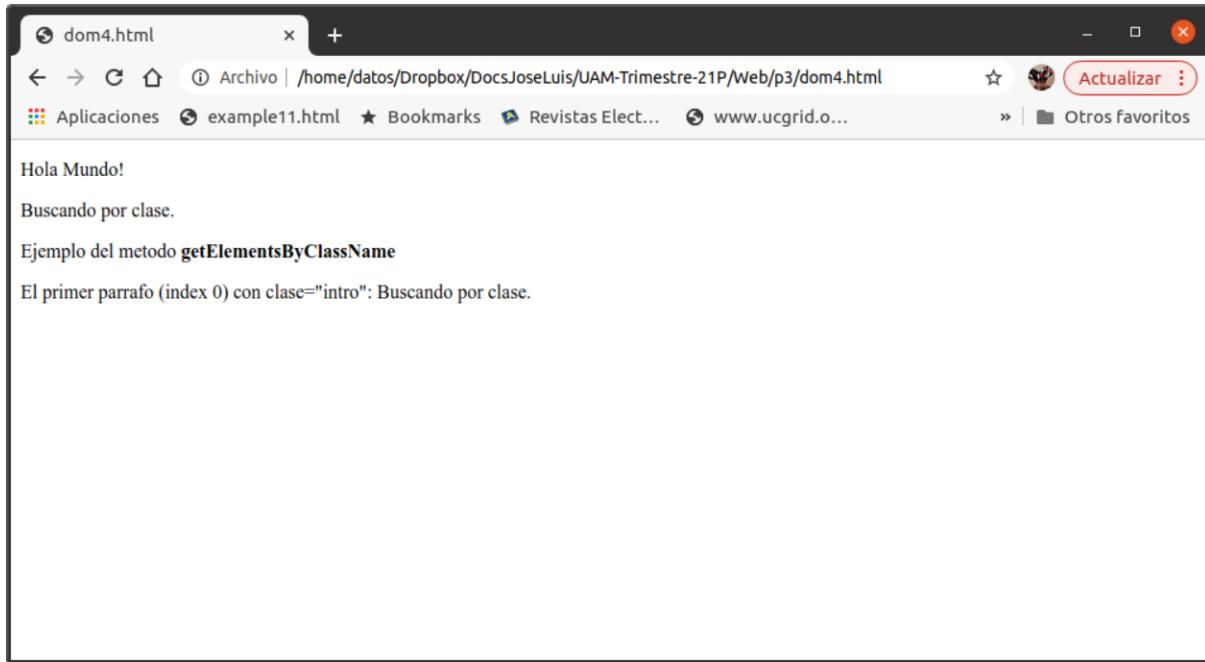
1 <!DOCTYPE html>
2 <html>
3   <body>
4
5     <p>Hola Mundo!</p>
6     <p class="intro">Buscando por clase.</p>
7     <p class="intro">Ejemplo del metodo <b>getElementsByClassName</b> </p>
8     <p id="demo"></p>
9     <script>
10       var x = document.getElementsByClassName("intro");
11       document.getElementById("demo").innerHTML = 'El primer parrafo (index
12         0) con clase="intro": ' + x[0].innerHTML;
13     </script>
14   </body>
</html>

```

Código 16: Encontrar elemento por clase

Salida:





1.8.4 Encontrando elementos selector CSS

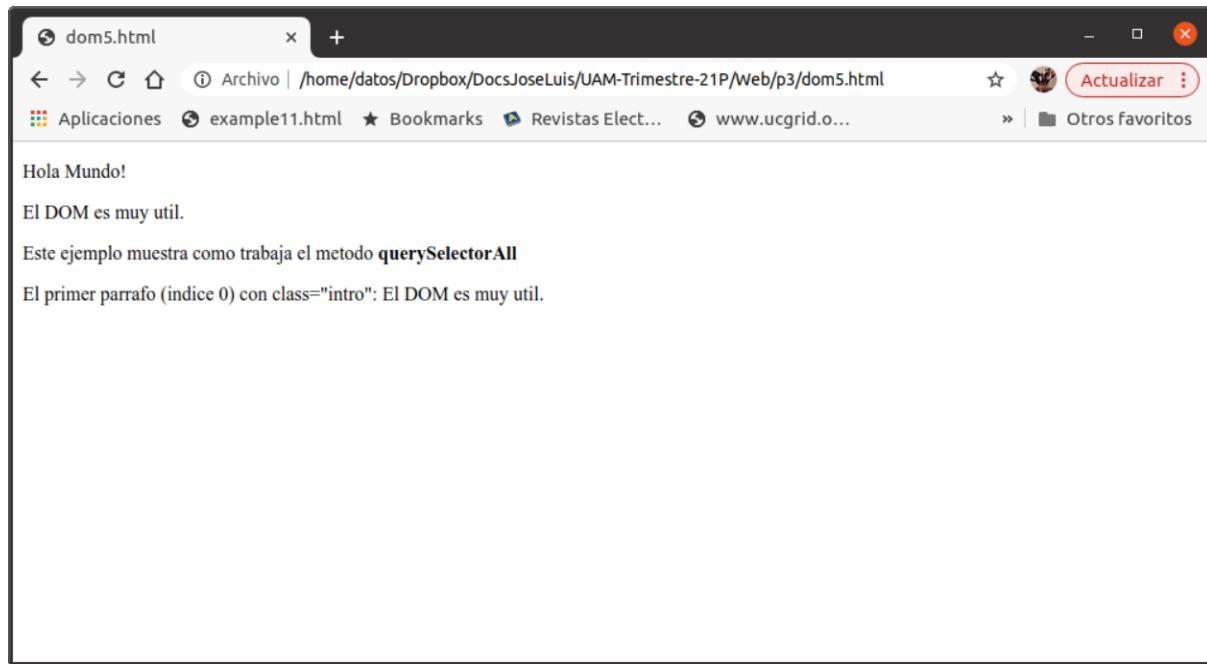
```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <p>Hola Mundo!</p>
5     <p class="intro">El DOM es muy util.</p>
6     <p class="intro">Este ejemplo muestra como trabaja el metodo <b>
7       querySelectorAll</b> </p>
8     <p id="demo"></p>
9     <script>
10       var x = document.querySelectorAll("p.intro");
11       document.getElementById("demo").innerHTML = 'El primer parrafo (indice
12         0) con clase="intro": ' + x[0].innerHTML;
13     </script>
14   </body>
15 </html>
```

Código 17: Encontrar elemento por selector

Salida:





1.8.5 Encontrando elementos de una colección form

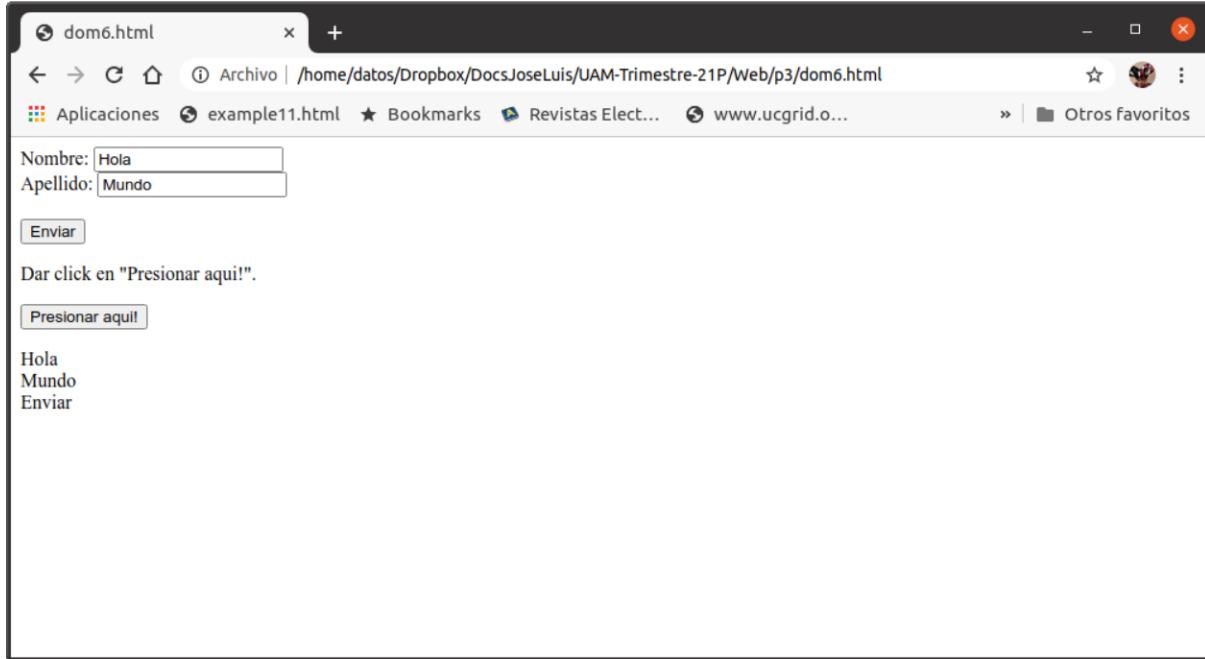
```

1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <form id="frm1">
6   Nombre: <input type="text" name="fnombre" value="Hola"><br>
7   Apellido: <input type="text" name="lnombre" value="Mundo"><br><br>
8   <input type="submit" value="Enviar">
9 </form>
10
11 <p>Dar click en "Presionar aqui!".</p>
12
13 <button onclick="getDatos()">Presionar aqui!</button>
14
15 <p id="demo"></p>
16
17 <script>
18 function getDatos() {
19   var x = document.forms["frm1"];
20   var text = "";
21   var i;
22   for (i = 0; i < x.length ;i++) {
23     text += x.elements[i].value + "<br>";
24   }
25   document.getElementById("demo").innerHTML = text;
26 }
27 </script>
28

```



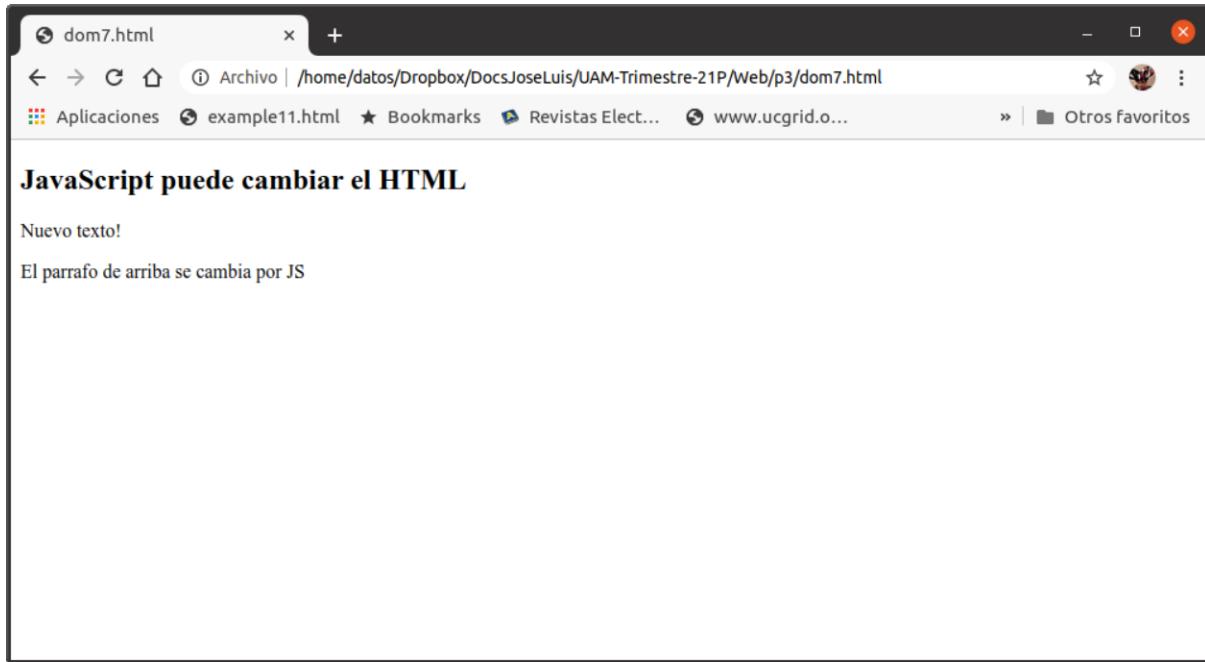
```
29 </body>
30 </html>
```

Código 18: Encontrar elementos de un formulario**Salida:****1.8.6 Cambiando el contenido de un elemento HTML**

La manera más fácil para modificar el contenido de un elemento HTML es usando la propiedad **innerHTML**.

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5   <h2>JavaScript puede cambiar el HTML</h2>
6
7   <p id="p1">Hola Mundo!</p>
8
9   <p>El parrafo de arriba se cambia por JS</p>
10
11  <script>
12    document.getElementById("p1").innerHTML = "Nuevo texto!";
13  </script>
14 </body>
15 </html>
```

Código 19: Modificando contenido de un elemento HTML**Salida:**



1.8.7 Cambiando el valor de un atributo

Para cambiar el valor de un atributo HTML se usa la sintaxis:

document.getElementById(id).attribute = nuevo valor

En el siguiente ejemplo se cambia el valor del atributo src de un elemento .

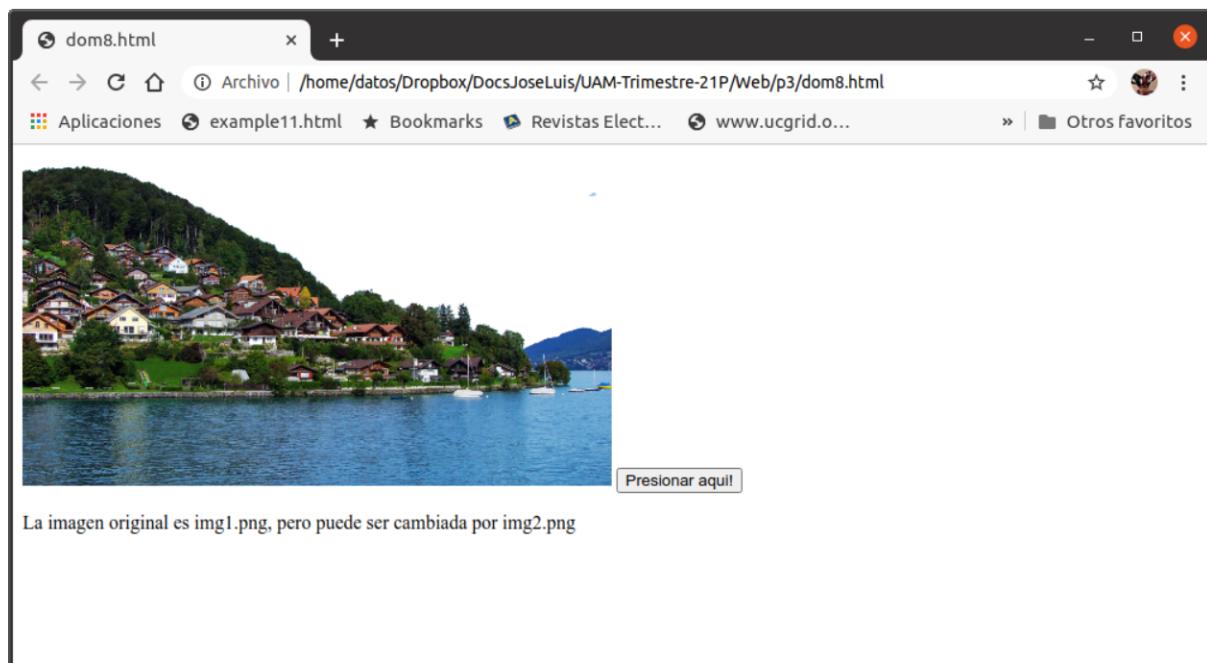
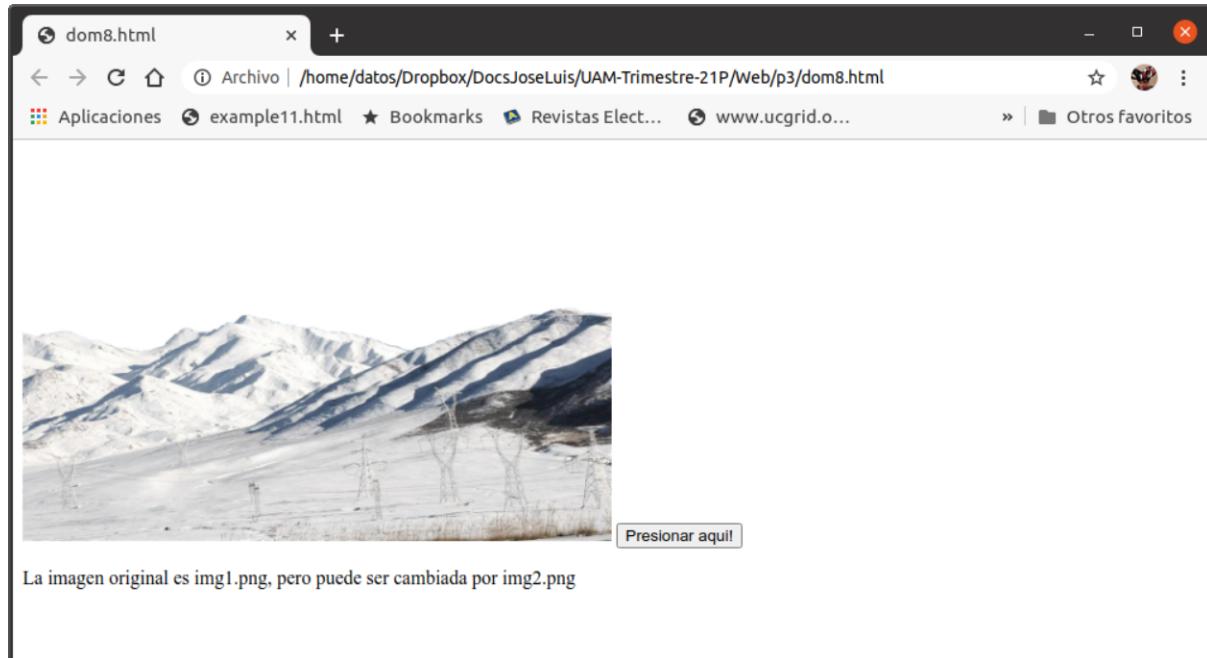
```

1  <!DOCTYPE html>
2  <html>
3  <body>
4
5      
6      <button onclick="setCambiar()">Presionar aqui!</button>
7
8      <script>
9
10     function setCambiar() {
11         document.getElementById("image").src = "img2.png";
12     }
13     </script>
14
15     <p>La imagen original es img1.png, pero puede ser cambiada por img2.png</p>
16
17 </body>
18 </html>
```

Código 20: Modificando el atributo de un elemento HTML

Salida:





Nota: para este ejemplo buscar dos imágenes y renombrarlas img1.png y img2.png respectivamente.

1.8.8 Cambiando un estilo

Para cambiar el estilo de un elemento HTML se usa la sintaxis:

```
document.getElementById(id).style.property = new style
```

En el siguiente ejemplo se cambia el estilo color del elemento identificado por **id1**.

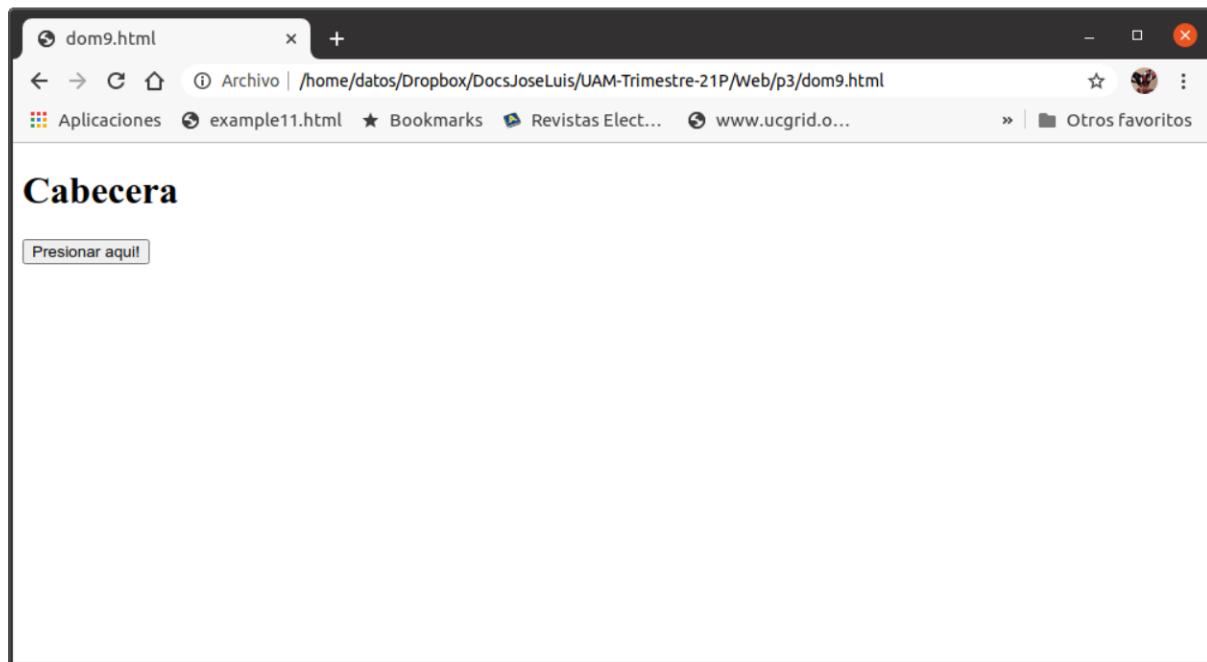
```
1 <!DOCTYPE html>
2 <html>
```

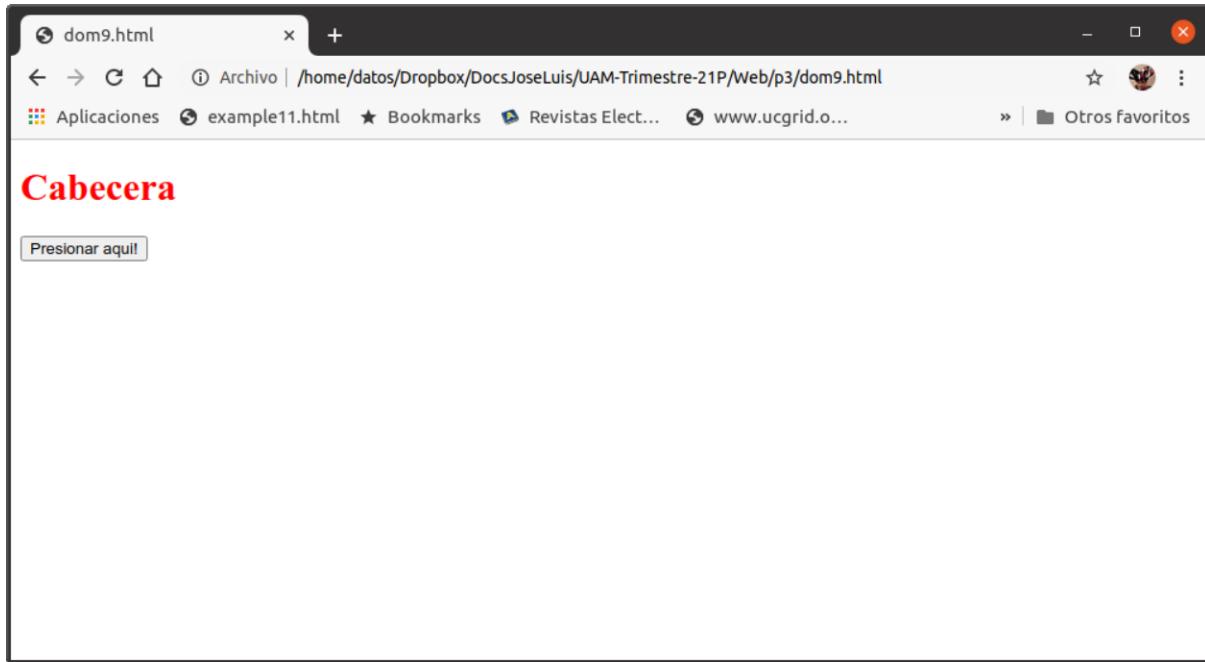


```
3 <body>
4
5   <h1 id="id1">Cabecera</h1>
6
7   <button type="button" onclick="setCambiar();">Presionar aqui!</button>
8   <script>
9
10  function setCambiar() {
11    document.getElementById('id1').style.color = 'red';
12  }
13 </script>
14 </body>
15 </html>
```

Código 21: Modificando el estilo de un elemento HTML

Salida:





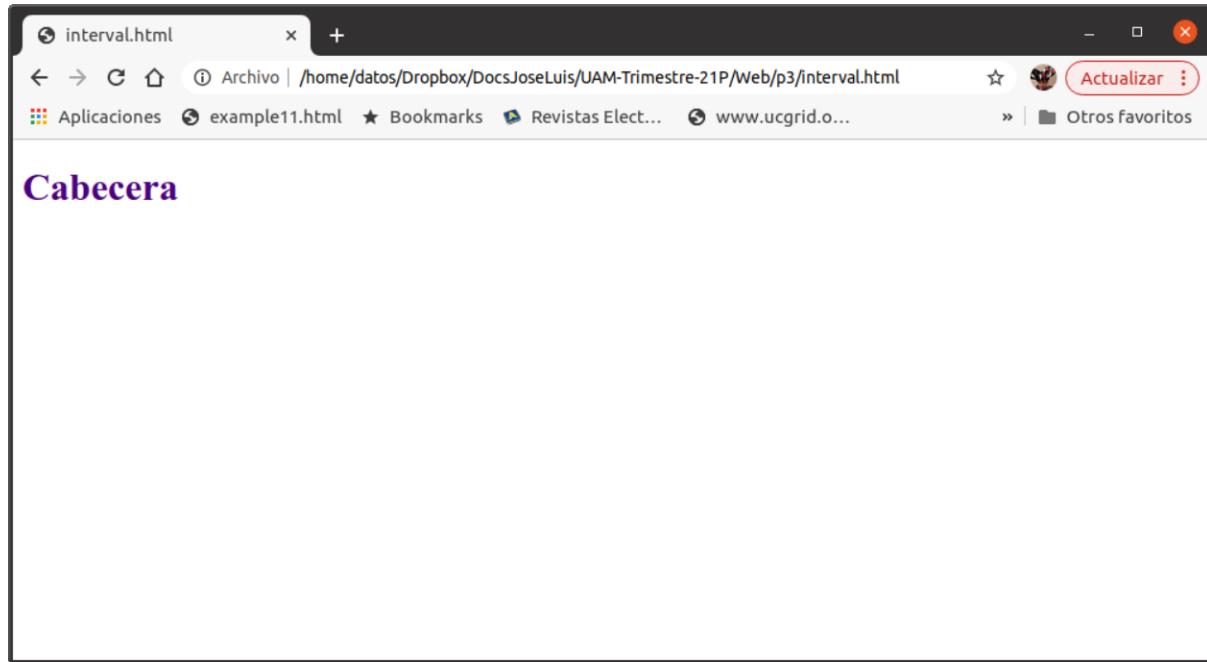
```

1 <!DOCTYPE html>
2 <html>
3 <body>
4
5   <h1 id="id1">Cabecera</h1>
6
7
8   <script>
9     var color = ["Blue ", "Green", "Red", "Orange", "Violet", "Indigo", "Yellow "
10    ];
11    var i=0;
12    function getRandomInt(max) {
13      return Math.floor(Math.random() * max);
14    }
15    function setCambiar() {
16      document.getElementById('id1').style.color = color[(i+1)%color.length];
17      i++;
18    }
19    setInterval(setCambiar, 3000);
20  </script>
21 </body>
22 </html>
```

Código 22: Modificando el estilo de un elemento HTML

Salida:





1.8.9 Ejercicio: Reloj

Implemente un reloj usando HTML, CSS y JavaScript. El segundero del reloj se debe mover de forma constante. Considere el siguiente código de ayuda.

```

1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <p>Hora:</p>
6
7
8 <p id="test"></p>
9
10 <script>
11
12     var d = new Date();
13     var dia = d.getDate();
14     var hora = d.getHours();
15     var minuto = d.getMinutes();
16     var segundo = d.getSeconds();
17
18
19     document.getElementById("test").innerHTML=hora+":"+minuto+":"+segundo;
20
21
22 </script>
23
24 </body>
```

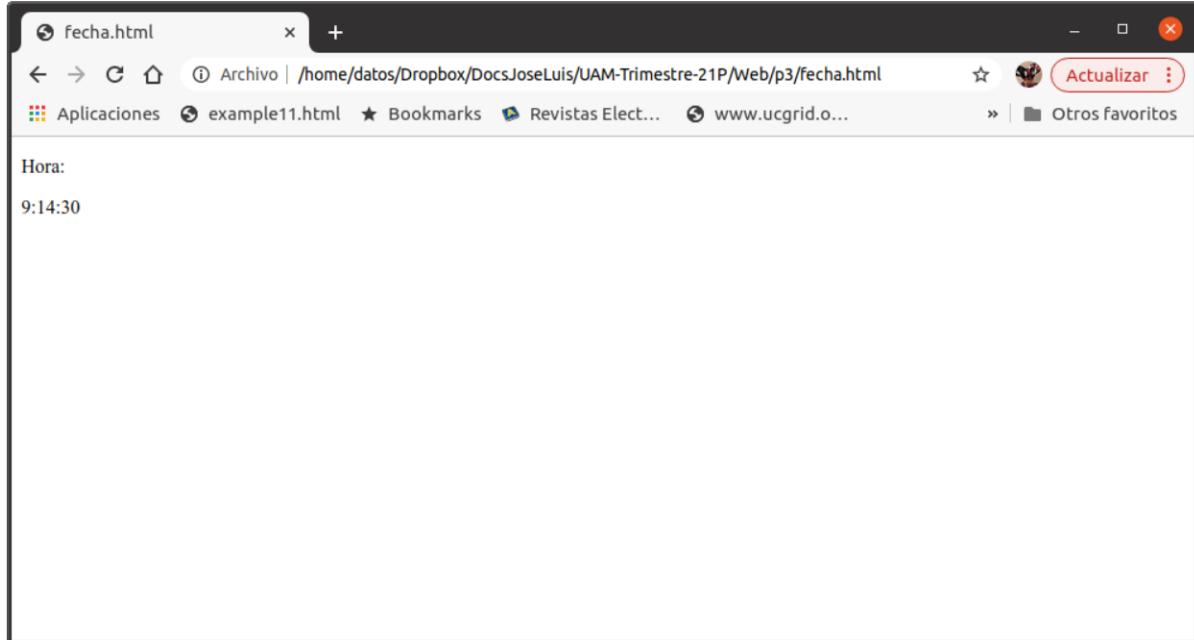


26

</html>

Código 23: Objeto Date en JavaScript

Salida:



Reloj esperado:



1.9 Eventos del teclado y del ratón

Por medio de JavaScript es posible controlar varios eventos del ratón, del teclado y del touch.

```

1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>Eventos del ratón</h1>
6
7 <p onclick="manejadorOnclik()" id="evento1">onclick</p>
8
9 <p oncontextmenu="manejadorOncontextMenu()" id="evento2">oncontextmenu</p>

```



```

10
11 <p onmousedown="manejadorMouseDown()" onmouseup="manejadorMouseUp()" id="evento3"
12   >onmousedown y onmouseup</p>
13
14 <p onmouseover="manejadorOnmouseover()" onmouseout="manejadorOnmouseout()" id=""
15   evento4">onmouseover</p>
16
17 <br>
18 <br>
19
20 <input type="text" onkeydown="manejadorOnkeydown(event)" onkeyup="
21   manejadorOnkeyup(event)" placeholder="onkeydown">
22
23 <br>
24 <p ontouchmove="manejadorOntouchmove(event)">Presiona touch y sin soltar
25   desplazate en el documento</p>
26
27 Coordenadas:<p id="test"></p>
28 <br>
29 <br>
30 Coordenadas:<p id="test2"></p>
31
32
33
34 <script>
35   function manejadorOnclik() {
36     document.getElementById("evento1").style.color = "red";
37   }
38
39   function manejadorOncontextMenu() {
40     document.getElementById("evento2").style.color = "green";
41   }
42
43   function manejadorMouseDown() {
44     document.getElementById("evento3").style.color = "lime";
45   }
46
47   function manejadorMouseUp() {
48     document.getElementById("evento3").style.color = "orange";
49   }
50
51   function manejadorOnmouseover() {
52     document.getElementById("evento4").style.color = "yellow";
53   }
54
55   function manejadorOnmouseout() {
56     document.getElementById("evento4").style.color = "black";
57   }

```



```

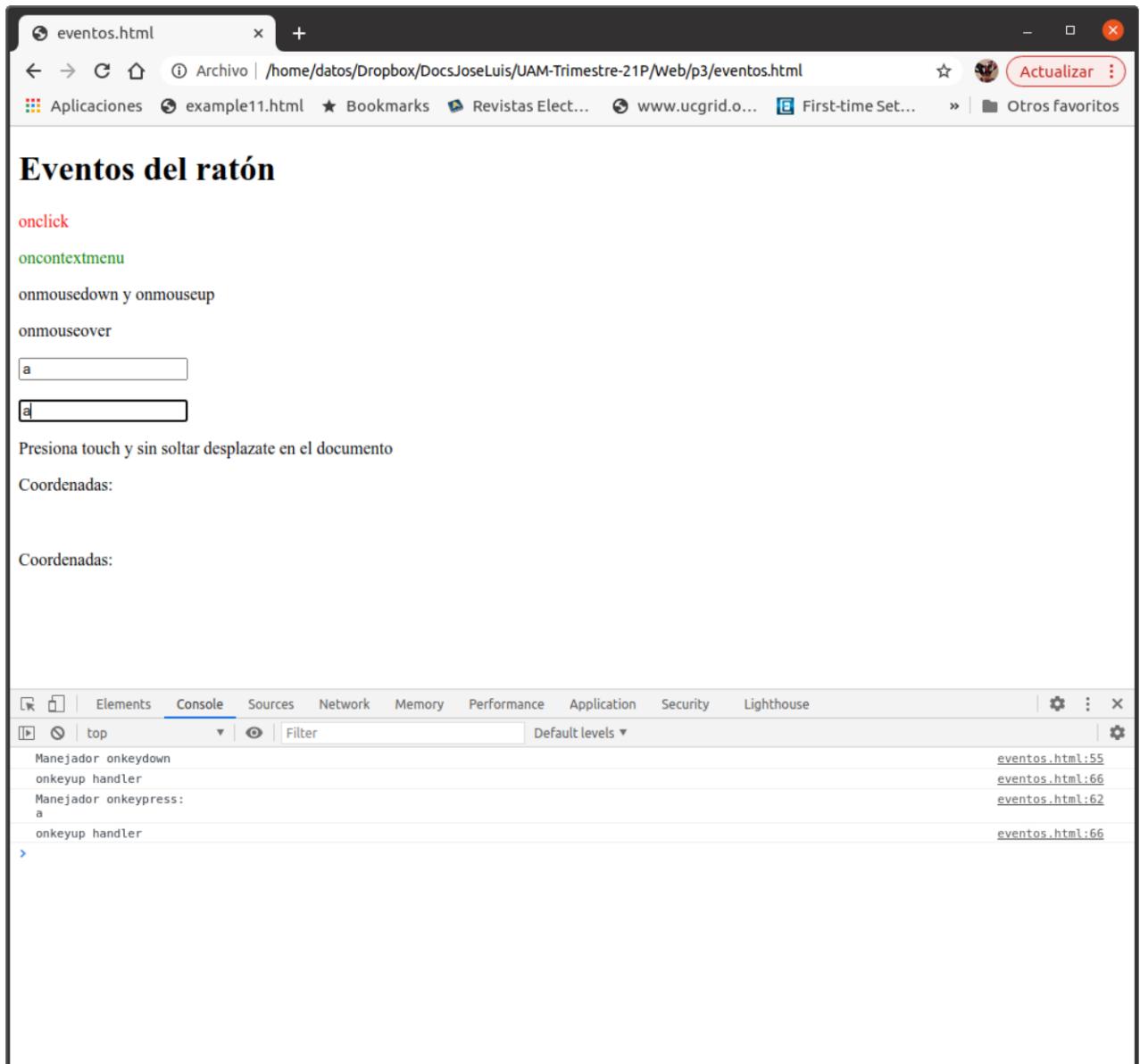
52 }
53
54 function manejadorOnkeydown(evt) {
55     console.log("Manejador onkeydown\n");
56
57 }
58 function manejadorOnkeypress(evt) {
59
60     var charCode = event.keyCode || event.which;
61     var charStr = String.fromCharCode(charCode)
62     console.log("Manejador onkeypress: \n" + charStr);
63 }
64
65 function manejadorOnkeyup(evt) {
66     console.log("onkeyup handler");
67 }
68
69 function manejadorOntouchmove(event) {
70     var x,z;
71     var y,w;
72     x = event.touches[0].clientX;
73     y = event.touches[0].clientY;
74     document.getElementById("test").innerHTML = x + " , " + y;
75     if(event.touches.length>1){
76         z = event.touches[1].clientX;
77         w = event.touches[1].clientY;
78         document.getElementById("test2").innerHTML = z + " , " + w;
79     }
80 }
81
82 </script>
83
84 </body>
85 </html>

```

Código 24: Eventos en JavaScript

Salida:





1.10 APIs HTML5

El alcance de Javascript ha sido expandido con un grupo de poderosas librerías accesibles a través una serie de APIs, como por ejemplo:

- Canvas** Esta API es una API de dibujo, específica para la creación y manipulación de gráficos. Utiliza métodos Javascript predefinidos para operar.
- Drag and Drop** Esta API hace que arrastrar y soltar elementos con el ratón en la pantalla sea posible también en la web.
- Geolocation** Esta API tiene la intención de proveer acceso a información correspondiente con la ubicación física del dispositivo que está accediendo a la aplicación. Puede retornar datos como la latitud y longitud utilizando diferentes mecanismos



4. **Web Storage** Esta API introduce dos atributos para almacenar datos en el ordenador del usuario: sessionStorage y localStorage. El atributo sessionStorage permite a los desarrolladores hacer un seguimiento de la actividad de los usuarios almacenando información que estará disponible en cada instancia de la aplicación durante la duración de la sesión. El atributo localStorage, por otro lado, ofrece a los desarrolladores un área de almacenamiento, creada para cada aplicación, que puede conservar varios megabytes de información, preservando de este modo información y datos en el ordenador del usuario de forma persistente.
5. **File** Este es un grupo de APIs desarrollada para proveer la capacidad de leer, escribir y procesar archivos de usuario.
6. **XMLHttpRequest Level 2** Esta API es una mejora de la vieja XMLHttpRequest destinada a la construcción de aplicaciones Ajax. Incluye nuevos métodos para controlar el progreso de la operación y realizar solicitudes cruzadas (desde diferentes orígenes).
7. **WebSockets** Esta API provee un mecanismo de comunicación de dos vías entre clientes y servidores para generar aplicaciones en tiempo real como salas de chat o juegos en línea.
8. **Web Workers** Esta API potencia Javascript permitiendo el procesamiento de código detrás de escena, de forma separada del código principal, sin interrumpir la actividad normal de la página web, incorporando la capacidad de multitarea a este lenguaje.
9. **Offline** Esta API apunta a mantener las aplicaciones funcionales incluso cuando el dispositivo es desconectado de la red.

1.11 Canvas

El elemento <canvas> es utilizado para dibujar gráficos en una página web mediante JavaScript. Canvas tiene varios métodos para dibujar trayectorias, rectángulos, círculos, texto, imágenes, etc. El elemento fue originalmente introducido por Apple en el OS X Dashboard y Safari. Internet Explorer, antes de la versión 9.0 beta, no admite de forma nativa <canvas>.

1.11.1 El área del canvas

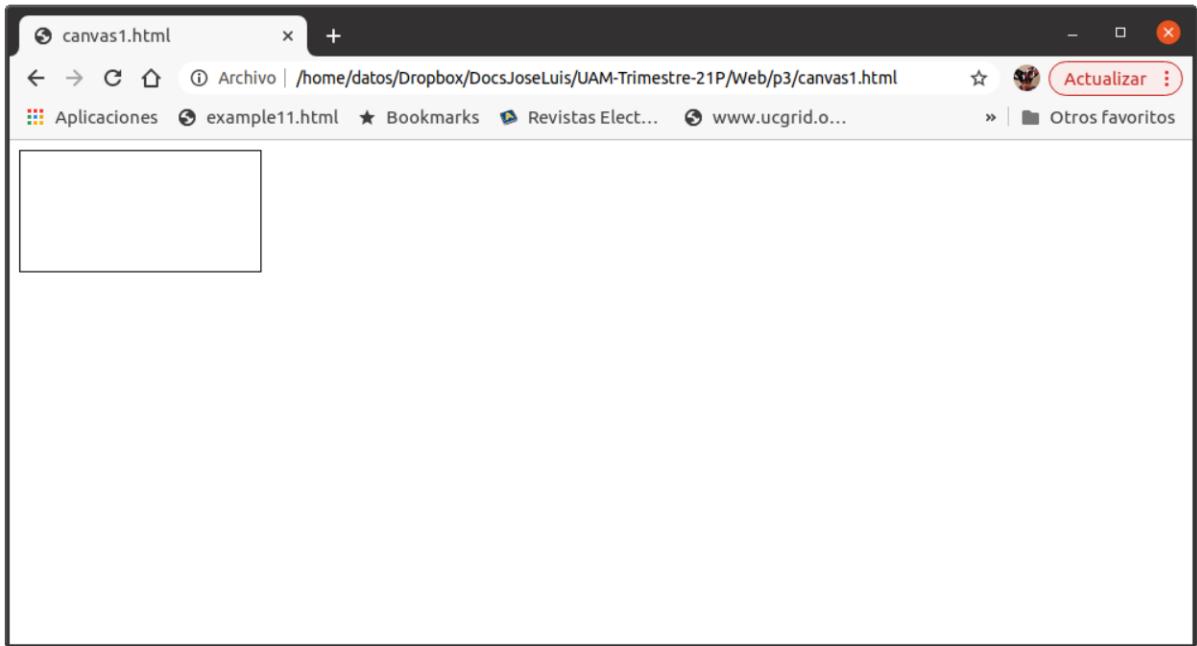
```

1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
6 Tu navegador no soporta canvas
7 </canvas>
```



```

8
9 </body>
10 </html>
```

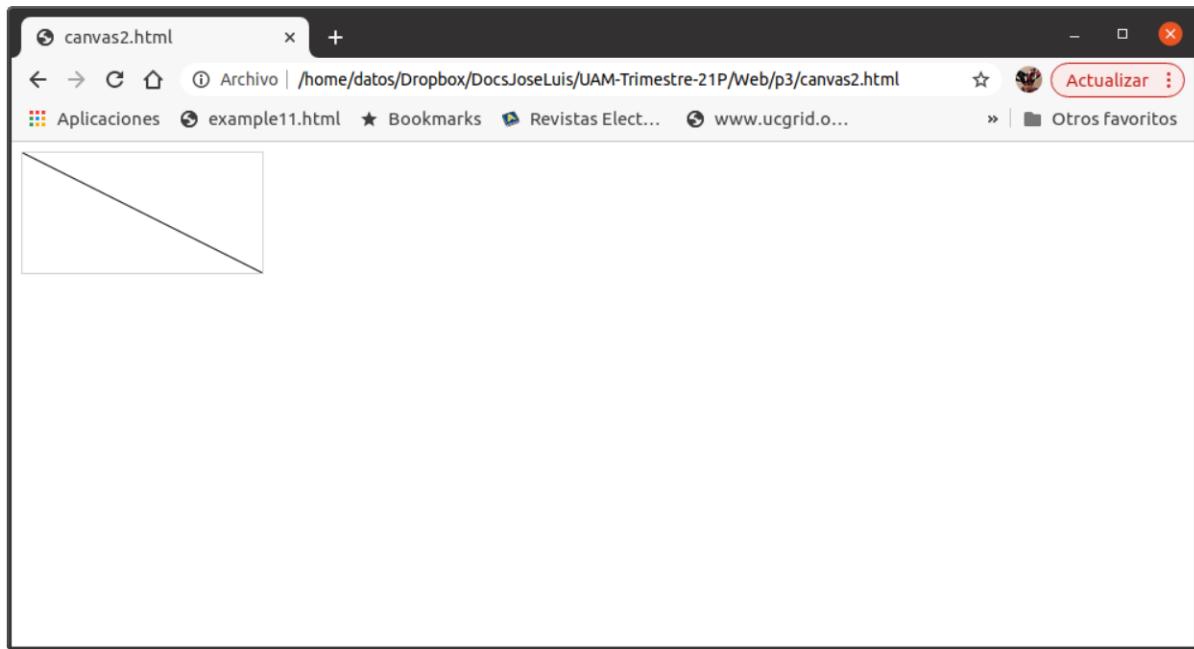
Código 25: Área del canvas**Salida:**

1.11.2 Una línea en canvas

```

1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <canvas id="canvas" width="200" height="100" style="border:1px solid #d3d3d3;">
6 Tu navegador no soporta canvas.</canvas>
7
8 <script>
9 var c = document.getElementById("canvas");
10 var ctx = c.getContext("2d");
11 ctx.moveTo(0,0);
12 ctx.lineTo(200,100);
13 ctx.stroke();
14 </script>
15
16 </body>
17 </html>
```

Código 26: Línea en canvas**Salida:**

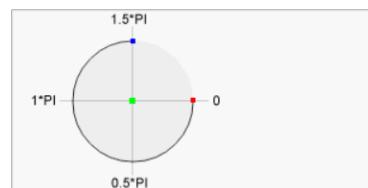


1.11.3 Un círculo en canvas

```

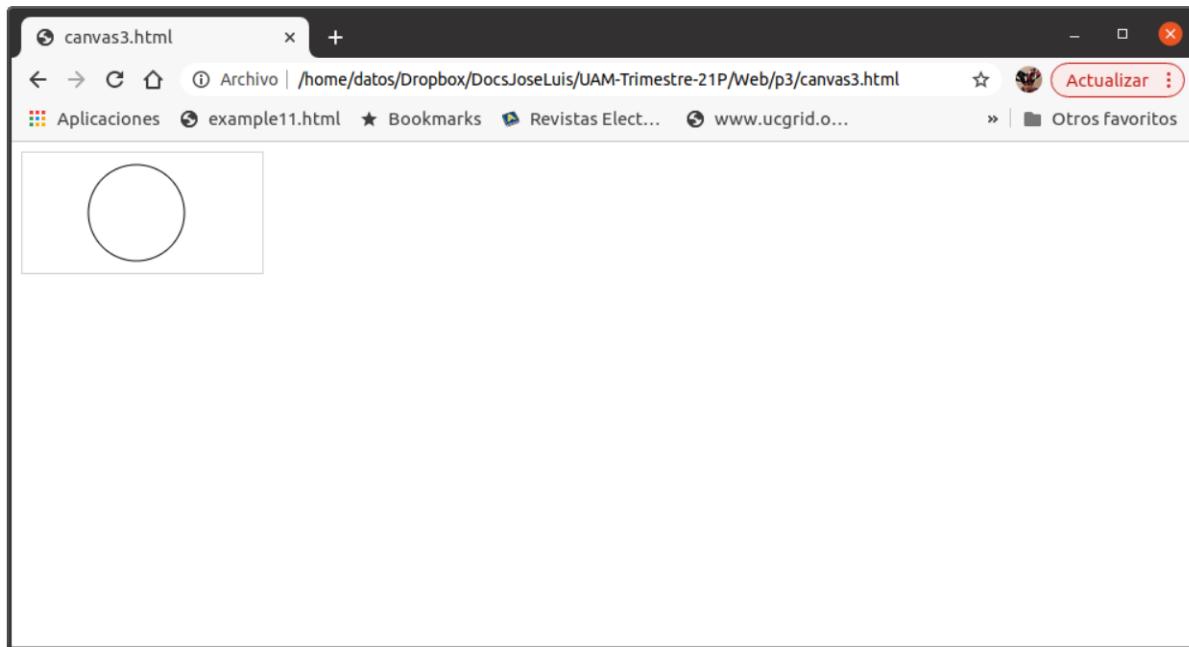
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <canvas id="canvas" width="200" height="100" style="border:1px solid #d3d3d3;">
6 El navegador no soporta canvas </canvas>
7
8 <script>
9 var c = document.getElementById("canvas");
10 var ctx = c.getContext("2d");
11 ctx.beginPath();
12 ctx.arc(95,50,40,0,2*Math.PI);
13 ctx.stroke();
14 </script>
15
16 </body>
17 </html>
```

Código 27: Circulo en canvas



Salida:





1.11.4 Una animación

Descargar fuente.

2 jQuery

jQuery es una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Basados en esta librería, existe una infinita cantidad de plugins (gratis y pagos) creados por desarrolladores de todo el mundo. Estos plugins resuelven situaciones concretas dentro del maquetado de un sitio, por ejemplo: un menú responsive, una galería de fotos, un carrousel de imágenes, un slide, un header que cambia de tamaño, el deslizamiento del scroll al hacer clic en un botón (anclas HTML), la transición entre páginas y miles de efectos más.

2.1 ¿Cómo se declara código jQuery?

Generalmente se descarga un archivo *.js que define la biblioteca jQuery y se inserta en la cabecera del documento HTML.

1 <!DOCTYPE html>



```

2 <html>
3   <head>
4     <script src="jquery-3.6.0.min.js"></script>
5     <script>
6       //Código JavaScript y jQuery
7     </script>
8   </head>
9   <body>
10
11   <h2>jQuery 3.6.0</h2>
12
13 </body>
14 </html>

```

2.2 Sintaxis jQuery

La sintaxis jQuery tiene la siguiente forma: `$(selector).accion()`, donde:

- `$`: Hace referencia a un acceso jQuery.
- `(selector)`: Selecciona/encuentra elementos HTML.
- `accion()`: Acciones que serán aplicada sobre los elementos seleccionados.

2.3 Un tipo de elemento como selector

En el siguiente ejemplo se define como selector el documento, el botón y los párrafos.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script src="jquery-3.6.0.min.js"></script>
5     <script>
6       $(document).ready(function(){
7         $("button").click(function(){
8           $("p").hide();
9         });
10      });
11     </script>
12   </head>
13   <body>
14
15   <h2>Cabecera</h2>

```



```
16  
17 <p>Párrafo 1.</p>  
18 <p>Párrafo 2.</p>  
19  
20 <button>Click para ocultar párrafos</button>  
21  
22 </body>  
23 </html>
```

Salida:



The image consists of two vertically stacked screenshots of a web browser window. Both screenshots show the same HTML content: a header titled "Cabecera", followed by two paragraphs ("Párrafo 1." and "Párrafo 2."), and a button labeled "Click para ocultar párrafos".

In the top screenshot, the button is visible and has a standard appearance. In the bottom screenshot, the button's background color has been changed to red, indicating it has been clicked.

2.4 Un elemento con id como selector

En el siguiente ejemplo se define como selector un elemento particular en base a su id.

```
1 | <!DOCTYPE html>
```



```
2 <html>
3 <head>
4 <script src="jquery-3.6.0.min.js"></script>
5 <script>
6 $(document).ready(function(){
7     $("button").click(function(){
8         $("#test").hide();
9     });
10 });
11 </script>
12 </head>
13 <body>
14
15 <h2>Cabecera</h2>
16
17 <p>Párrafo 1.</p>
18 <p id="test">Párrafo 2 con id test.</p>
19
20 <button>Ocultar párrafo 2</button>
21
22 </body>
23 </html>
```

Salida:



The image displays two side-by-side screenshots of a web browser window titled "ejemplo3.html". Both screenshots show the same HTML content:

```
<h2>Cabecera</h2>


Párrafo 1.



Párrafo 2 con id test.



Ocultar párrafo 2


```

In the top screenshot, the paragraph "Párrafo 2 con id test." is visible. In the bottom screenshot, this paragraph has been hidden by a CSS rule triggered by a mouse event. The browser's toolbar and address bar are visible at the top of both screenshots.

2.5 Manejo de eventos

2.5.1 Controlando el evento mouseup.

```
1 | <!DOCTYPE html>
```



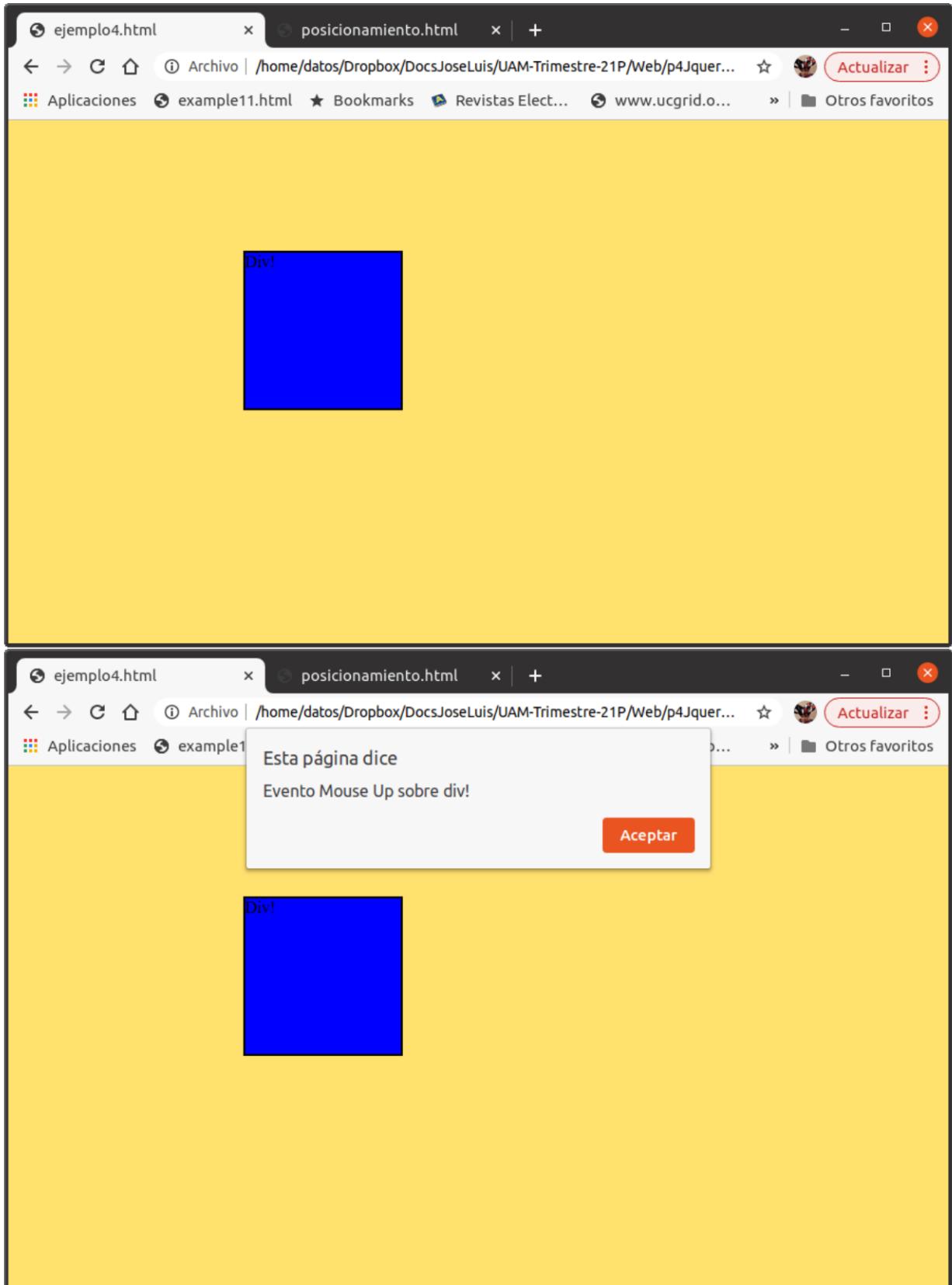
```

2 <html>
3 <head>
4 <style>
5 html, body {
6   background: #FFE16E;
7   width: 100%;
8   height: 100%;
9   margin: 0;
10  padding: 0;
11 }
12
13 .azul{
14   width:150px;
15   height:150px;
16   position: relative;
17   background:blue;
18   left: 25%;
19   top: 25%;
20   border: 2px solid black;
21 }
22
23 </style>
24 <script src="jquery-3.6.0.min.js"></script>
25 <script>
26 $(document).ready(function(){
27   $("#miDiv").mouseup(function(){
28     alert("Evento Mouse Up sobre div!");
29   });
30 });
31 </script>
32 </head>
33 <body>
34
35 <div class="azul" id="miDiv">
36   Div!
37 </div>
38
39 </body>
40 </html>

```

Salida:





2.5.2 Entrando y saliendo de un elemento.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
```



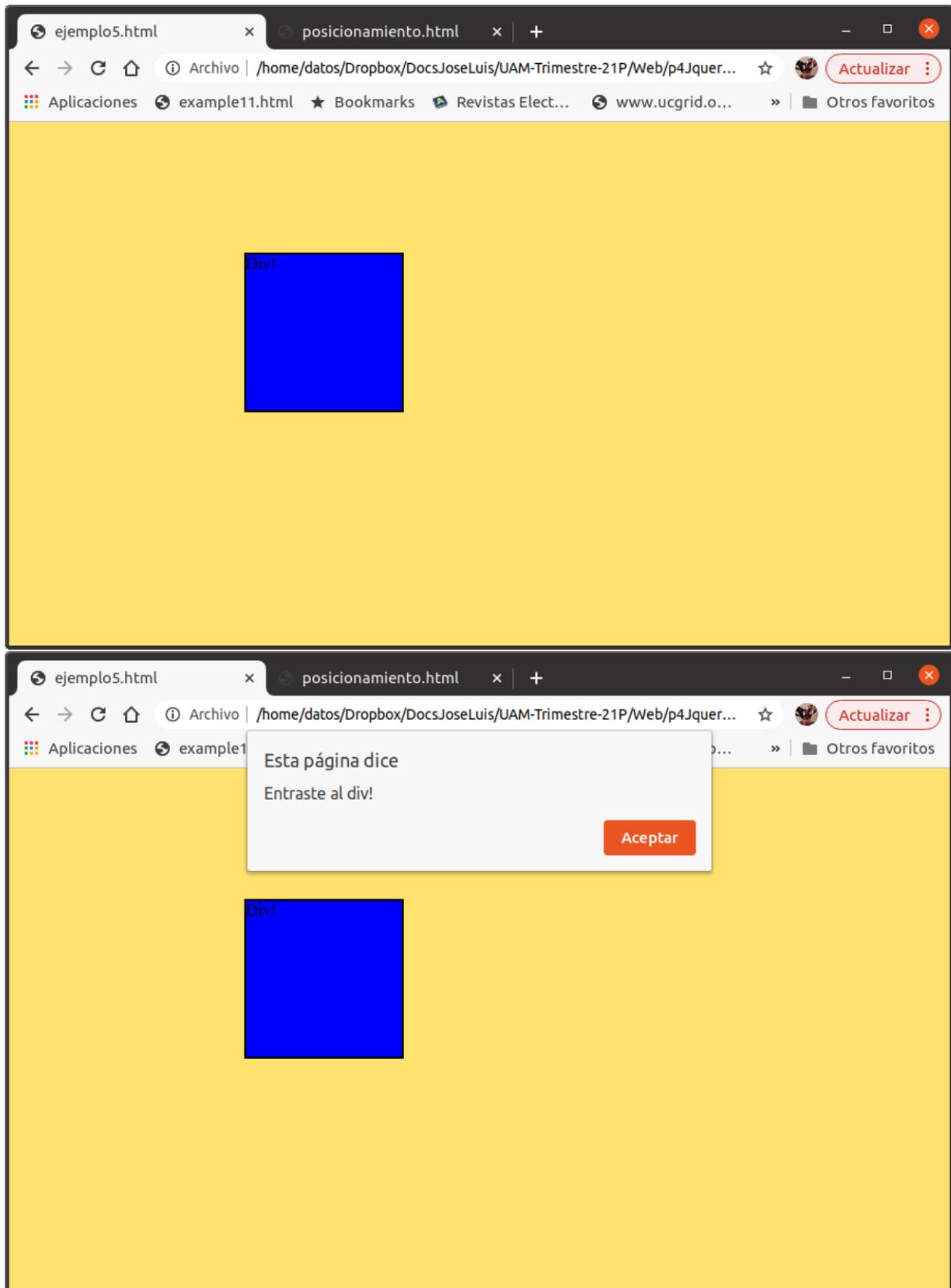
```

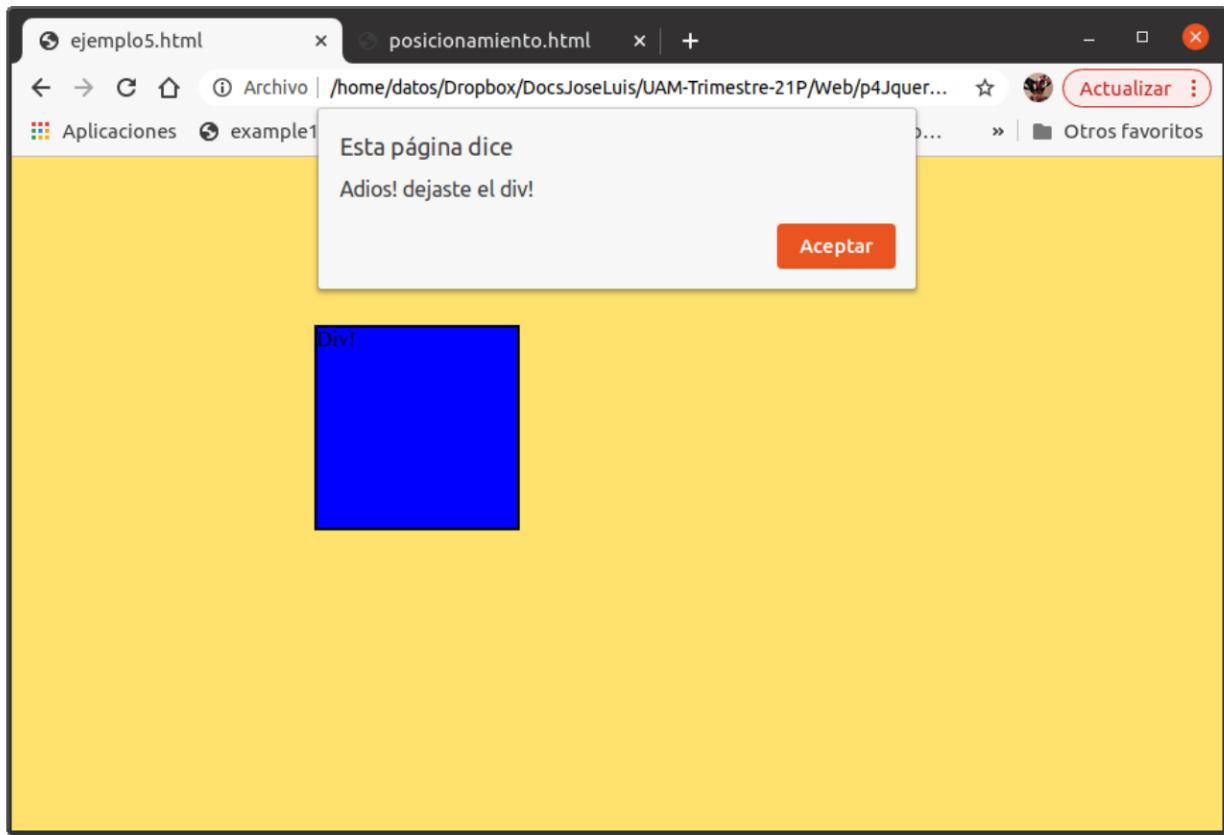
4 <style>
5 html, body {
6     background: #FFE16E;
7     width: 100%;
8     height: 100%;
9     margin: 0;
10    padding: 0;
11 }
12
13 .azul{
14     width:150px;
15     height:150px;
16     position: relative;
17     background:blue;
18     left: 25%;
19     top: 25%;
20     border: 2px solid black;
21 }
22
23 </style>
24 <script src="jquery-3.6.0.min.js"></script>
25 <script>
26 $(document).ready(function(){
27     $("#miDiv").hover(function(){
28         alert("Entraste al div!");
29     },
30     function(){
31         alert("Adios! dejaste el div!");
32     });
33 });
34 </script>
35 </head>
36 <body>
37
38 <div class="azul" id="miDiv">
39     Div!
40 </div>
41
42 </body>
43 </html>

```

Salida:







2.5.3 Múltiples eventos.

```

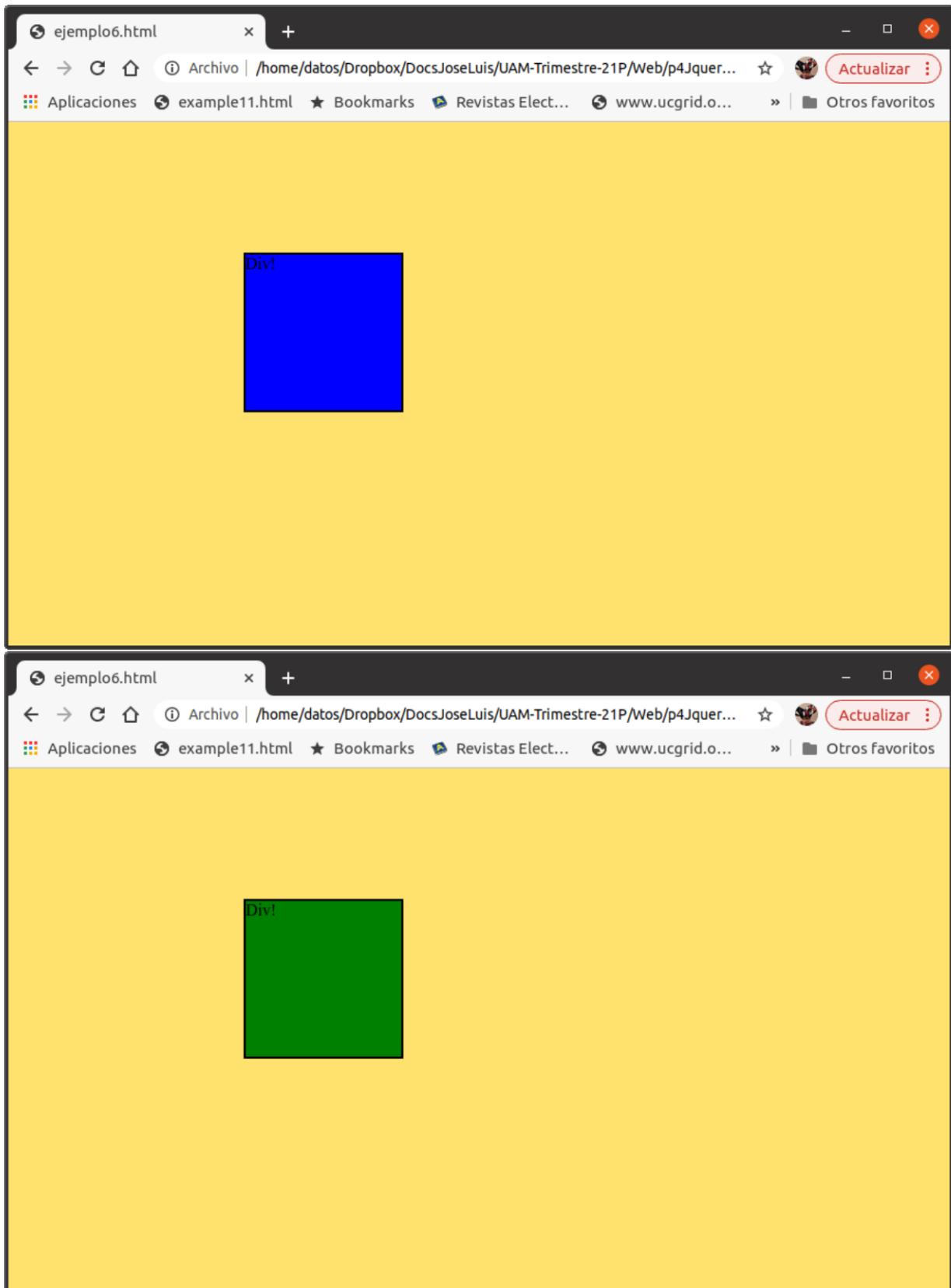
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <style>
5 html, body {
6   background: #FFE16E;
7   width: 100%;
8   height: 100%;
9   margin: 0;
10  padding: 0;
11 }
12
13 .azul{
14   width:150px;
15   height:150px;
16   position: relative;
17   background:blue;
18   left: 25%;
19   top: 25%;
20   border: 2px solid black;
21 }
```

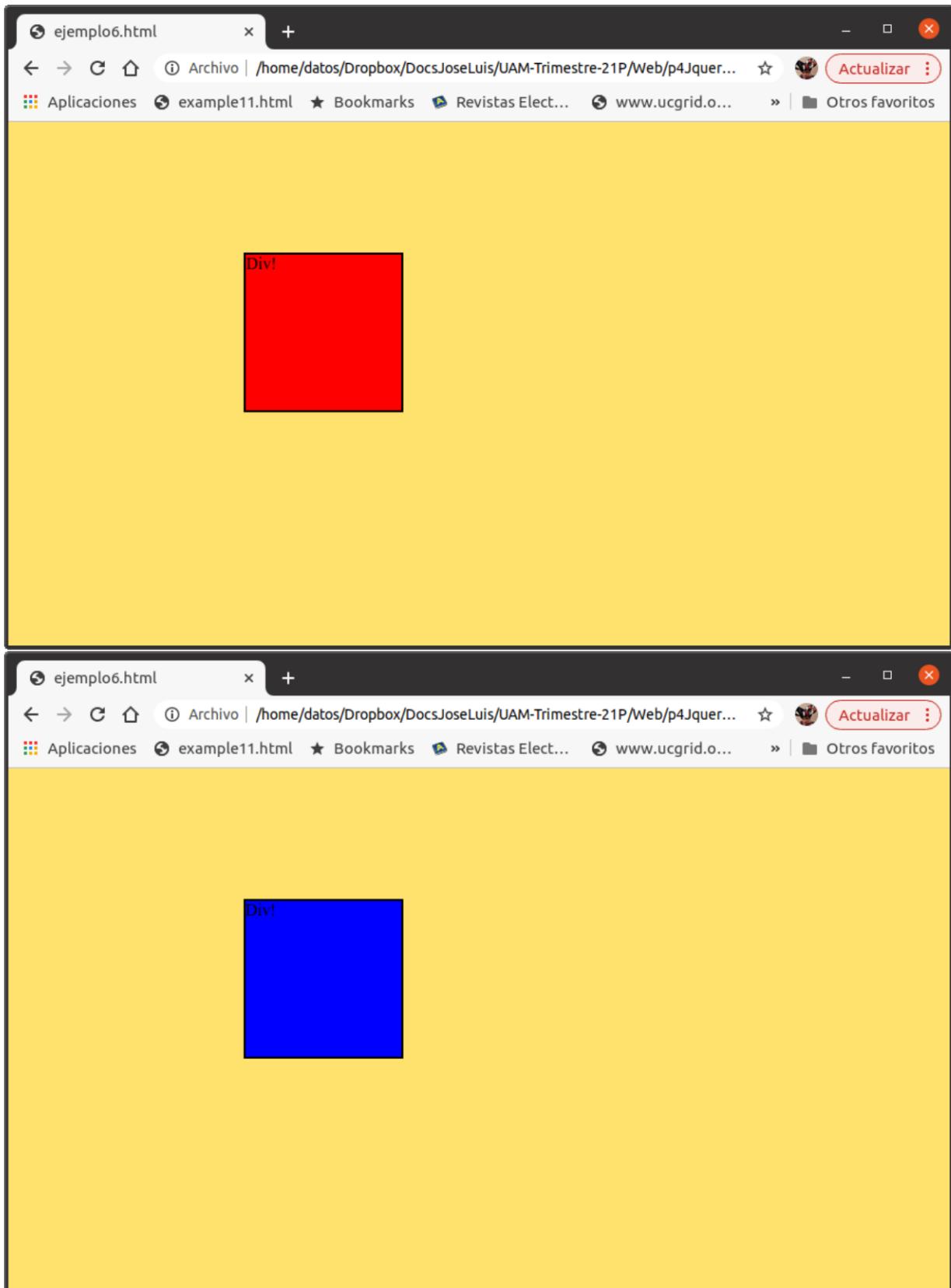


```
22
23 </style>
24 <script src="jquery-3.6.0.min.js"></script>
25 <script>
26 $(document).ready(function(){
27     $("#miDiv").on({
28        mouseenter: function(){
29             $(this).css("background-color", "green");
30         },
31        mouseleave: function(){
32             $(this).css("background-color", "red");
33         },
34         click: function(){
35             $(this).css("background-color", "blue");
36         }
37     });
38 });
39 </script>
40 </head>
41 <body>
42
43 <div class="azul" id="miDiv">
44     Div!
45 </div>
46
47 </body>
48 </html>
```

Salida:







2.6 Animación.

Es posible aplicar una animación fácilmente mediante jQuery.

```
1 <!DOCTYPE html>
2 <html>
```



```
3 <head>
4 <style>
5 .clase{
6     background:red;
7     height:100px;
8     width:100px;
9     top:200px;
10    position:absolute;
11 }
12
13 </style>
14 <script src="jquery-3.6.0.min.js"></script>
15 <script>
16 $(document).ready(function(){
17     $("button").click(function(){
18         $("#div1").animate({left: '250px'});
19
20         $(".clase").animate({
21             left: '250px',
22             opacity: '0.5',
23             height: '150px',
24             width: '150px'
25         });
26
27         $("#div3").slideUp();
28
29     });
30 });
31 });
32 </script>
33 </head>
34 <body>
35
36 <button>Iniciar animación</button>
37
38 <p>Por default, todos los elementos HTML tienen una posición estática, y
39 no pueden moverse. Para manipular la posición, recuerden primero
40 aplicar la propiedad posición, por medio de CSS, a relative, fixed o
41 absolute.
42
43 </p>
```



```
41 <div style="background:blue;height:100px;width:100px;position:absolute;"  
42   id="div1"></div>  
43 <div class="clase" id="div2"></div>  
44 <div style="background:green;height:100px;width:100px;top:302px;position:  
45   absolute;" id="div3"></div>  
46 </body>  
47 </html>
```

Salida:





2.7 Agregar nuevos elementos.

En jQuery para agregar un nuevo elemento se utiliza **append**.

```
1 | <!DOCTYPE html>
```



```
2 <html>
3 <head>
4 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.
min.js"></script>
5 <script>
6 $(document).ready(function(){
7     $("#btn1").click(function(){
8         $("div").append(" <b>Párrafo nuevo</b><br>");
9     });
10
11     $("#btn2").click(function(){
12         $("ol").append("<li>Item nuevo</li>");
13     });
14 });
15 </script>
16 </head>
17 <body>
18
19 <div>
20 <p>Este es un párrafo.</p>
21 <p>Este es otro párrafo.</p>
22 </div>
23 <ol>
24     <li>item 1</li>
25     <li>item 2</li>
26     <li>item 3</li>
27 </ol>
28
29 <button id="btn1">Insertar texto</button>
30 <button id="btn2">Insertar a lista</button>
31
32 </body>
33 </html>
```

Salida:



